# SECTION 1 - OVERVIEW

## 1.1 INTRODUCTION

This technical report supports the objective in Task 3 of the Joint Operations Planning and Execution System (JOPES) Database Migration Technical Proposal, FY95, revised 24 August 1995, under contract number DCA100-94-D-0016, " . . . [to] prototype distributed processing as a means of supporting efficient database access and provide a written report of the results". It proposes transitioning the Global Command and Control System (GCCS) from the closed Worldwide Military Command and Control System (WWMCCS) mainframe to a more open client/server environment. It specifically embraces the Government's desire to seek Commercial Off-the-Shelf (COTS) technology solutions wherever possible. This report discusses the use of the Open Systems Foundation's (OSF) Distributed Computing Environment (DCE) as a method for migrating first generation GCCS database applications to a more distributed environment.

There are two main sections in this technical report. The first is a summary describing the process and technical analysis that Systems Research and Applications (SRA) Corporation used to develop a Defense Infrastructure Initiative (DII) DCE prototype. The DII DCE prototyping effort provides the basis for migrating existing DII database applications into a more distributed client/server computing environment through the use of Transarc's DCE, a COTS middleware product, and Transarc's Encina, a COTS transaction monitor product.

The second section contains the detailed analysis of the DCE prototyping effort, including requirements analysis, derivation of hardware and software architectures, a description of the DCE prototype operating environment, product analysis of DCE and Encina including requirements satisfaction, performance testing and product limitations, a proposed migration strategy, and overall conclusions and recommendations.

## 1.2 SUMMARY

The DCE prototype's purpose is to demonstrate the feasibility of incorporating distributed computing and transaction processing capabilities into DII database applications using Transarc's DCE and Encina products. The first step was to derive requirements which constituted the various functional and non-functional capabilities for distributed computing in the DII environment. Then, an existing candidate application, the Add Carrier thread of the Scheduling and Movement (S&M) application, was targeted for prototyping to determine if DCE and Encina could fulfill the derived requirements. A general software migration strategy was developed and implemented to provide for migrating an existing application into a three-tiered client/server architecture. Also, a generic DCE and Encina cell architecture was derived and implemented based on the specific network topology of SRA's prototyping facilities. Finally, the end product was analyzed and tested to verify that the requirements were satisfied. The results of this analysis are presented in detail in the second section of this paper along with recommendations and conclusions concerning a general migration strategy and the outline for a pilot project to develop and field a DCE based application in the DII environment.

### 1.2.1 Requirements

Prior to evaluating the capabilities of both DCE and Encina, a number of functional, non-functional, and application-specific requirements were identified. Functional requirement areas included distributed resource management, compliance with the existing and future GCCS database permissions schemas, transaction management, and data retrieval. Non-functional requirement areas included cross-platform and environment independence, product interoperability, usability and administration, security, performance, scalability, fault tolerance, and compatability with existing GCCS applications. Application-specific requirements for implementing the Add Carrier thread of S&M included adding air, land, and sea cargo carrier data, retrieving Geographic Location (GEOLOC) data, and facilitating operational plan (OPLAN) data updates and retrieval across multiple resource managers. Driving all of these requirements was the improvement of user functionality, performance, and administration requirements present in the current DII database applications through the capabilities provided by a distributed computing environment.

### 1.2.2 Architecture and Operational Environment

The DCE prototype incorporates a three-tiered client/server architecture that uses DCE as the fundamental infrastructure to provide a flexible distributed computing model. The three-tiered model provides several advantages (including increased performance, increased scalability, and others) over the traditional two-tiered model used by database applications today in DII for distributed systems. Before beginning the prototyping effort, a migration strategy was derived for converting a two-tiered application into a three-tiered application independent of implementation details pertaining to specific applications. The DCE and Encina products provided the framework for this strategy through migration of the two-tiered S&M application to a three-tiered client/server architecture.

The distributed computing architecture consists of DCE and Encina cell architectures which are dependent upon the network topology at a given site. In this prototype a single DCE and Encina cell were configured over a three server network consisting of both local-area network (LAN) and wide-area network (WAN) network connections. The DCE prototype was implemented and tested solely on SUN servers running the Solaris 2.3 Operating System (OS).

Encina Transactional Remote Procedure Calls (TRPCs) and DCE remote procedure calls (RPCs) were used as the method of communication between the first and the second tiers in the three-tiered client/server architecture. Use of this technology allowed clients and servers to be located on different machines in the network. This feature provided the ability to perform both transactional database updates and non-transactional database queries while minimizing network traffic and maximizing the distribution of application processing between multiple machines. Additionally, through the implementation of an OPLAN Directory Service (ODS), the user had the ability to select from available distributed resources when updating an OPLAN, providing a user friendly and flexible distributed system..

### 1.2.3 Product Analysis

Transarc's DCE and Encina products were used to implement the prototype based on several factors including their adherence to open systems standards, support for the UNIX environment, product interoperability, conformance to functional and non-functional requirements, and consistency with the DII's Common Operating Environment (COE). Although all candidate DCE products should mirror OSF's DCE, Transarc has developed a number of additional value-added utilities that simplify DCE application development and administration for UNIX-based systems making it an appropriate choice for use in the DCE prototype. DCE alone does not provide transaction management capabilities; a separate DCE-compliant transaction processing system was needed to satisfy transaction management requirements.

The completed DCE prototype, using the tools available in Transarc's DCE and Encina products, was analyzed with respect to functional, non-functional, and application-specific requirements. Overall, Transarc's DCE product used in conjunction with Encina satisfies or exceeds almost all of the requirements as shown in Table 1-1. Specifically, the satisfaction of the scalability and performance improvement requirements show the success of the three-tiered architectural approach used in the DCE prototype. A DCE only solution was only able to satisfy a few requirements without an additional software development effort for necessary functionality, showing the need for a value added product such as Encina.

Transarc and Oracle corporations, in future releases of their products, claim to fulfill requirements not completely satisfied in the DCE prototype. However, some points need to be emphasized. The DCE environment, like any distributed system, is inherently difficult to administer due to the significant number of synchronous and asynchronous processing components accessing multiple objects within the networked environment. The script based administrative solutions provided by DCE and Encina provided administration capabilities, but with a high level of effort. Future versions of Encina that provide graphical administrative capabilities could improve this process. Also, existing technologies such as Graphical User Interfaces (GUIs) and Relational Database Management System (RDBMS) products must use industry standard interfaces to be able to freely interact with DCE and Encina. Given this caveat, other value added products, automated tools, and utilities to build applications and administer the DCE and Encina environment are still necessary to provide the most flexible solution in the future.

Specific performance tests compared remote database query and update operations using the DCE prototype versus the existing two-tiered S&M application. The DCE prototype performed as well or better than the existing S&M application in all test cases, and significantly outperformed the existing S&M application for all database operations involving large data retrievals over low bandwidth networks.

*Table 1-1: Requirements Satisfaction Results for DCE Prototype.*

| Requirement Area | DCE | DCE with Encina |
|---|---|---|
| Distributed Resource Management | ✗ | ✓+ |
| Transaction Management Requirements | ✗ | ✓ |
| Data Retrieval Requirements | ✓- | ✓ |
| GCCS Current Database Permissions Scheme | ✗ | ✓+ |
| GCCS Future Database Permissions Scheme | ✗ | ✗ |
| Cross-Platform Independence | ✓ | ✓ |
| Product Interoperability | ✓ | ✓ |
| Usability and Administration | ✓- | ✓- |
| Security | ✓- | ✓- |
| Performance | ✗ | ✓+ |
| Scalability | ✓ | ✓ |
| Fault Tolerance | ✗ | ✓ |
| Existing GCCS Database Application Implementations | ✓- | ✓- |
| Application Specific Requirements | ✓- | ✓ |
| Overall Rating | ✓- | ✓ |

| | |
|---|---|
| ✗ | Requirement Not Satisfied |
| ✓- | Requirement Partially Satisfied |
| ✓ | Requirement Satisfied |
| ✓+ | Requirement More Than Satisfied |

### 1.2.4 Conclusions and Recommendations

The DCE prototyping effort provided an opportunity to investigate the implications of incorporating Transarc DCE's middleware component and Encina's transaction manager into a DII application. It demonstrates the ability to migrate the application from the existing two-tiered architecture into a three-tiered client/server architecture, allowing for greater distribution and scalability of processing components. This architecture ensures higher performance and resource manager availability, while reducing data access times. The following conclusions can be drawn from this effort :

- Transarc's DCE and Encina products should be candidates for inclusion into the DII COE.

- A product analysis of application development environments and GUIs must be undertaken. A candidate product for developing distributed computing applications using DCE and Encina must produce POSIX compliant threads which can interact with DCE Remote Procedure Calls (RPCs) and Encina TRPCs. Existing technologies in the GCCS COE such as OIS's ScreenMachine and Sybase's Gain Momentum products do not provide this capability.

- A DCE pilot project will not only incorporate additional S&M functionality to use distributed computing, but will also deploy the DCE application at specific GCCS installation sites. This will afford the opportunity to provide extensive testing of DCE and also validate and refine other efforts investigating the introduction of DCE into the DII COE, including the *GCCS DCE Implementation Plan* and the *Distributed Computing Environment (DCE) Applications Programming Guide* developed by DISA's Distributed Computing Working Group (DCWG).

# SECTION 2 - ANALYSIS

## 2.1 INTRODUCTION

This section presents a detailed analysis of Transarc's DCE and Encina products used in the DII DCE prototyping effort. This effort is intended to provide the basis for migrating future DII applications to an open, distributed client/server computing environment. The objective of the DCE prototype is initially presented, followed by a description of functional and non-functional requirements for distributed computing in DII, and application specific requirements. A synopsis of the architecture and the operating environment provides a comparison between a two-tiered and a three-tiered client/server architecture as well as a description of the DCE prototype application software. Also, the ability of using DCE as well as the combination of DCE and Encina to satisfy requirements is analyzed based upon the results of the DCE prototyping effort. In addition, comparisons of performance tests between the prototype and original application are summarized. Finally, product benefits and limitations are addressed and a future migration strategy and general conclusions are presented.

### 2.1.1 Purpose and Objectives

The DCE prototype demonstrates the feasibility of incorporating DCE and transaction processing capabilities (through the use of existing COTS software) into DII applications. A thread of S&M, a GCCS database application, was selected as the application to prototype for this effort. S&M implements a two-tiered client/server architecture, where the user interface and business logic have been combined in the first tier, and the resource manager, an ORACLE RDBMS, is incorporated in the second tier. The DCE prototype demonstrates the ability to migrate the application into a three-tiered architecture, separating the user interface and business logic into two separate tiers, and the resource manager into a third tier. Using the three-tiered architecture allows the application to take advantage of several distributed computing features, such as increased performance and increased scalability, which cannot be achieved in the original two-tiered design.

One objective of the DCE prototype was to simulate a subset of operations that were performed by a GCCS database application. The DCE prototype implements this objective by incorporating the functionality of the Add Carrier thread of S&M. It provides the ability to add several database records to a database as a transactional update, query data from a database, and incorporates a directory service to facilitate updates and data retrieval across multiple resource managers.

Another objective was to evaluate COTS products as the method of implementing distributed computing into the DII environment. Transarc's DCE and Encina transaction management COTS products were selected and evaluated based on their compliance with open systems standards, support for the UNIX environment, and consistency with the DII's COE.

The final objective was to develop a migration strategy for GCCS database applications. This was performed in support of the work being done by DISA's DCWG to prepare for the inclusion of DCE into the GCCS 3.x baseline. This task also supports parallel efforts within the DCWG to develop artifacts to prepare application developers to incorporate DCE into both existing and first generation database applications.

### 2.1.2 Assumptions

The DCE prototyping effort was based on the following assumptions :

- The DCE prototype assumes the use of DCE as the product of choice to perform this migration and does not investigate other distributed computing options such as Common Object Request Broker Architecture (CORBA).
- The evaluation of COTS products (specifically Transarc's DCE 1.0.3a individually and in combination with Transarc's Encina 1.1) as a complete solution to the distributed operating environment of DII database applications. DCE will not be evaluated with other compatible products, such as administration tools, nor will any other versions of DCE or Encina be evaluated. However, any shortcomings or version dependent properties that affect the application in future product releases will be analyzed and discussed.

## 2.2 REQUIREMENTS

In order to accomplish the goal of introducing distributed computing into the DII environment using DCE, a number of functional, non-functional, and application specific requirements must be satisfied. The functional areas include: distributed resource management, support for the current and future GCCS database permissions schema, transaction management, and data retrieval. Non-functional areas include: cross-platform and environmental independence, product interoperability, usability and administration, security, scalability, fault tolerance, and support for current GCCS database application implementations. Application-specific requirements address issues pertaining to the specific application selected for implementation in the DCE prototype. The following sections discuss each of these functional, non-functional, and application-specific areas.

### 2.2.1 Functional Requirements

In order to implement a distributed database application in the DII environment using DCE, several functional requirements must be satisfied. Driving these requirements is replication of existing user functionality found in GCCS database applications as well as the current and future GCCS database permissions schemas and implementation. To support a true migration strategy, it is important to satisfy all functional requirements in order to provide the user with at least the same level of functionality that is supported by the current applications.

**2.2.1.1 Distributed Resource Management Requirements.** Using a combination of ORACLE environment variables and the SQL*Net transportation protocol, existing GCCS database applications allow the user, at installation time, to select a target database on which operations should be performed. In this instance, the user can only change databases by modifying start-up scripts or shell environment variables. Both of these methods were outside the scope of the application in GCCS 2.x and could only be accomplished through a non-programmatic manual process. Consequently, a requirement is to allow the user to select a database programmatically from within the application. Subsequent database requests are then routed to the appropriate database Resource Manager (RM), which may be local or remote. The RM in turn performs the appropriate operations on the selected database. The location of the RM should be transparent to the user.

**2.2.1.2 GCCS Database Permissions Schemas.** In the current release of GCCS (Version 2.0), the permissions scheme for accessing information in the GCCS Core database consists of the following permission types:

- OS level permissions which consist of read, write, and execute privileges on system files. These permissions are associated with UNIX User IDs and control access to various GCCS executables.

- ORACLE table permissions based on the ORACLE User ID. These permissions control access to GCCS Core Database tables based on the user's role (i.e., a user can either access all, none, or a portion of the GCCS Core Database tables based on their assigned role).

- User OPLAN permissions based on OPLAN series permissions and OPLAN specific permissions. These two permissions together determine whether a user has access to an OPLAN based on both GCCS User ID and the particular OPLAN that is being accessed. They are implemented via the application accessing ORACLE tables to determine if the user has access to the OPLAN based on GCCS User ID and OPLAN availability.

- Functional permissions that define the functions a user may perform on a particular OPLAN. Much like user OPLAN permissions above, they are implemented via the application accessing an ORACLE table to determine what functional permissions are associated with the user.

In future releases of GCCS (3.x and beyond), the permissions scheme for accessing information from the GCCS Core database will consist of the following types of permissions:

- OS level permissions and ORACLE table permissions as previously described.

- User OPLAN permissions and functional permissions enforced on all data accesses of any GCCS Core Database table. Although these are the same permissions used in the current scheme, they will be implemented via ORACLE views which use the user's ORACLE ID as well as access control information from other ORACLE tables to verify data access privileges.

Any database application in the DII environment is required to be compatible with and enforce both database permissions schemas upon any requests for data by the user.

**2.2.1.3 Transaction Management Requirements.** A database application in the DII environment must have the ability to provide transactional updates to local and remote databases. A transactional update may consist of several insertions, updates, and/or deletions to and from the database and is executed as a single business rule or transaction. If any portion of the transaction fails, then any operations that have already been processed are rolled back, thus returning the database to its original state. This action results in a single monatomic operation. The application programmer should be able to identify a transaction's boundaries in the business logic code and should not be required to insert explicit commits and rollbacks; instead, the database RM should accomplish these tasks. In the event of any failure or error, the application must rollback the database to its previous state and log the appropriate error message to the user. Additionally, in the event of a catastrophic failure, the system should be able to restore database transactions from the last checkpoint or savepoint in order to ensure overall system integrity.

**2.2.1.4 Data Retrieval Requirements.** A database application in the DII environment must provide two basic functions in the area of data retrieval from a database query. First, it must provide a mechanism for the user to retrieve data from local or remote databases. Secondly, it must provide an interface to display the retrieved data in an organized and efficient manner. Additionally, in the event of a fatal error during a database query, the system must return to a stable state so that the query can be retried. Finally, multiple users should be able to query the same data simultaneously.

### 2.2.2 Non-Functional Requirements

Non-functional requirements of a database application in the DII environment must support the functional requirements of the system. Some of these requirements consist of measurable capabilities of the system such as performance and fault tolerance. Other issues such as product interoperability, the ability of the application to function across multiple hardware platforms in a multitude of environments, and the ability to administer a system effectively are also crucial for a system to be fielded in the DII COE.

**2.2.2.1 Cross-Platform and Environment Independence.** Cross-platform and environment independence allows all components of the application to function within a heterogeneous, distributed system among multiple hardware platforms and environments. Current database applications must be able to operate using the Solaris 2.x OS. Additionally, it is desired that these applications operate under the 9.x HP-UX OS as well. The applications must also be able to update an ORACLE7 database, the current DII COE Database Management System (DBMS). Any migration plan, however, must include products and methods which allow the application to support an increasing list of hardware platforms, operating systems, and database products. In addition, access to distributed system components on dissimilar hardware platforms and operating environments should be transparent to the user and in effect, should perform exactly as if the components were running together on the same local machine.

**2.2.2.2 Product Interoperability Requirements.** A database application in the DII environment must provide basic interoperability and compatibility with other software products used within a heterogeneous distributed computing environment by incorporating standard product interfaces and conforming to open systems standards. Such standards such as X/Open and ODBC for database operations will allow the most flexibility and interaction between products.

**2.2.2.3 Usability and Administration Requirements.** Sufficient administrative and operational support tools and utilities are required in order to simplify the use, administration, and maintenance of a database application in the DII environment as well as ensure the continuous operation of a fielded production system. In particular, the system should provide a qualified system administrator with the tools to configure, install, administer, upgrade, and de-install a distributed system.

Configuration and installation of a database application should be completed at segment installation time, and should be performed through an automated process. This process should encompass installing the software, configuring the system to recognize and incorporate the software, and initializing the software into an operational state.

At a minimum, the same administrative level of effort and personnel should be required as is necessary to support the current DII COE. The software should be able to recover automatically from some errors without interaction from the system administrator. It is also required that the administrator be able to easily monitor and diagnose problems with the system.

The system requires the ability to upgrade components from a configuration management perspective. Software releases should be backwards compatible (i.e., a user should be able to still use Release 1.0 software, even if Release 2.0 software has been installed). Additionally, the upgrade process should also be automated with minimal interaction required by the administrator. Finally, de-installation should also be an automated process which removes all application components from the system.

**2.2.2.4 Security Requirements.** Security requirements to the system initially include authentication and authorization when logging into the system. The system must authenticate the user and verify the user s authorization level prior to allowing the user to perform operations on the system. The system should require the user to authenticate only once to the system. Access is granted to the user based on groups or roles determined by the user's identity. Unauthorized access during login should result in an appropriate error message as well as notification to the system. Unauthorized access during an attempted database operation should result in an error message logged to the user and no operation being performed. If properly authenticated, the user should be authorized to perform a finite set of business operations on the database using DII applications based on the specific functional rules of that application.

**2.2.2.5 Performance Requirements.** One of the primary requirements of a database application in the DII environment is to provide improved or at least comparable performance over an existing distributed database application. Performance improvements over existing database applications should especially be seen over networks with limited bandwidth approaching 64 kilobits per second (kbps) when accessing remote data over these networks.

**2.2.2.6 Scalability Requirements.** A database application in the DII environment should be scalable, i.e., it should distribute processing across all available resources without respect to physical boundaries. This should be minimally accomplished by distributing presentation layer and business logic processing across multiple machines in order to reduce large memory and resource requirements currently necessary for database application servers. The system should also provide a method of load balancing processes between application servers to minimize bottlenecks and provide more throughput for the application.

**2.2.2.7 Fault Tolerance Requirements.** In order to mitigate risk and cost, the system is required to maintain a certain level of fault tolerance ensuring reliability, maintainability, and availability. Reliability guarantees that all applications, infrastructure, value-added and support products must be able to perform their intended functions for a specified duration, free of operational failure. Maintainability is attained by preserving a specific system state or restarting a system to a definite condition by using specific procedures and resources. Availability is achieved by providing a significant degree of operable and committable states during application component interaction, network communication, and transaction processing.

Continuous system availability and reliability is required during optimal and sub-optimal run-time conditions. Optimal conditions provide for full software and hardware functionality within the operational system and include all infrastructure, applications, system resources, and resource managers inherent within the system. Sub-optimal conditions include partial functionality within the system or unavailability of various system components and resources, communication, infrastructure, or resource managers. In order to provide continuous system availability and reliability, adequate redundancy of system functional components and resources is necessary. In the event that any system component or resource is unavailable, sufficient replication and redundancy should provide secondary functionality and resource availability. By incorporating adequate redundancy, the system can remain in a continuous operational state during system, infrastructure, application upgrades, and secondary installations.

**2.2.2.8 Existing GCCS Database Application Implementations.** For minimal re-engineering and true application migration, it is desired that any migration of GCCS database applications be accomplished without having to greatly modify existing code. Most existing GCCS database applications can be divided into three major logical parts:

- A GUI which manages the presentation layer of data for the user.
- Business logic implemented in a 3rd Generation or 4th Generation (3GL/4GL) programming language. The business logic is made up several business rules, each of which consist of several sequences of operations which together defines an operation of work in the system.

.

- A database update mechanism to update information on a local or remote database.

Some existing database applications and the corresponding technologies used to implement them are shown in Table 2-1.

*Table 2-1: Existing GCCS Database Application Technologies.*

| Application Name | GUI/Presentation Layer | Business Logic | Database Update Mechanism |
|---|---|---|---|
| Scheduling & Movement<br>System Services<br>Ad Hoc Query | Screen Machine v. 1.4.1<br>(Generated Ada Source) | Ada & C (3GL) | Pro*Ada (UPI)<br>SQL*NET |
| Requirements Data Analysis<br>Non-Unit Personnel Generator | Gain Momentum v. 1.0 | GEL (4GL) | Gain Proxy Server<br>(OCI like calls) |

Various other application suites also exist for GCCS Database Applications which present additional requirements for a complete migration strategy which can encompass all of GCCS.

### 2.2.3 Application Specific Requirements

In order to demonstrate full functionality migrating an existing GCCS database application into a truly distributed computing environment, the functional requirements described above must be present in a candidate application to prototype.

The S&M GCCS database application thread Add Carrier met the functional requirements derived from current database applications and was targeted for migration. The programmatic distributed resource management requirement did not exist in the original application, as the user was only allowed to connect to a single database per application session to handle database requests. Therefore, new functionality was added to the DCE prototype to satisfy this "transparent" database access requirement.

**2.2.3.1 Add Carrier Requirements.** The Add Carrier business rule requirements are available in the original JOPES S&M Software Requirements Specification (SRS). For the purpose of this prototype the following requirements should be emphasized:

- Ability to add air, land, or sea cargo/Passenger (Pax) Carriers to the GCCS Core Database based on user selection and inputs.

- Implementation of minimal data validation rules for the GUI interface including validation of correctly typed data (integers only in integer field, etc.), verification of valid dates, verification of required not-null fields, and verification of sequential itineraries.

- Execution of business logic for Add Carrier including insertion of records into the following GCCS Core Database tables:

  - Oplan_Carrier (Carrier/OPLAN pair)
  - Carrier (Carrier information)
  - Carrier_Itinerary (Up to 28 Itinerary Records based on user input)
  - Send_Queue (SCHDET & SCHPET transactions).

- Ability to update a local or remote database transparent to the user.

The Add Carrier requirement satisfies the transaction management and GCCS database permissions schema requirements of the DCE prototype.

**2.2.3.2 Geographic Location Lookup Requirements.** All GEOLOC fields in the Add Carrier application should have a field-level help function allowing the user to query for GEOLOCs according to the JOPES S&M SRS. The user should have the ability to search on a variety of GEOLOC data fields as well as by Country State Code. The query constructed should be an outer join of the Smdb_Geographic_Location and Smdb_Country_State tables in the GCCS Core Database. Paging should also be available to the user as well as a slide bar to reference a specified row number. Multiple queries should be allowed on the same screen. Upon selection of a row, the GEOLOC code should be copied to the original screen and the help screen should be closed.

The GEOLOC Lookup functionality satisfies the data retrieval requirement of the DCE prototype.

**2.2.3.3 OPLAN Directory Service Requirements.** The DCE prototype requirement for allowing the user to easily select which remote database to perform database operations on infers the concept of an ODS. In the existing GCCS Core Database, OPLANs are networked to multiple database sites with information stored in the Oplan_Routing database table. Amongst these sites, data for a particular OPLAN is synchronized. An update to data on any of these sites is propagated to all other sites on the distribution list to maintain synchronization. This is accomplished in GCCS 2.0 using Transaction Distribution Service (TDS), but in the future could be replaced with a multitude of methods (including database replication).

An operation of work in existing database applications against the GCCS Core Database is usually in support of a single OPLAN. Therefore, before performing work on that OPLAN, the user is required to select a database (essentially a site) where that OPLAN is resident and available. Functionally, this site will usually be the local database, but in the case of database failure or a site without a local database, can also be remote. After selecting this site and OPLAN, all database operations will then be performed against the selected database.

ODS should be implemented as a query search and should allow the user to search based on OPLANs, Database Server Name, or DCE Cell Name. ODS should also provide paging ability if the number of records retrieved is larger than the screen display. ODS should also provide the ability to perform work on multiple OPLANs per user session by re-entering the ODS screen before performing work on a new OPLAN.

## 2.3 ARCHITECTURE AND OPERATING ENVIRONMENT

This section discusses different aspects of the DCE prototyping architecture and operating environment. Specifically, this section first provides a general description of client/server architectural considerations for any environment including two-tiered and three-tiered architectures. This description is followed by a general migration strategy for converting an existing two-tiered application into a three-tiered application. The DCE prototype s specific architecture is then presented. Finally, the DCE prototype s operating environment is discussed.

### 2.3.1 General Client/Server Architectural Considerations

In order to achieve a true heterogeneous open, distributed system, it is necessary to first understand general client/server architecture principals before a client/server architecture can be selected for a specific application or project. There are several factors that influence client/server architectures. First and foremost, COTS packages providing a consistent interface infrastructure (such as Transarc's DCE) must be evaluated for usability in a client/server architecture. This assures maximum compatibility and interoperability between software and hardware components. Also, the use of COTS packages (such as Transarc's Encina) provide the ability to easily incorporate additional capabilities such as transaction management into the client/server architecture.

Other general factors to consider include:

- Accessibility to multiple heterogeneous resource managers within the constraints of a single transaction.
- Use of multiple heterogeneous server applications within the context of concurrent processing.
- Capability to provide fault tolerance through the use of component redundancy and transaction rerouting during system fault detection and recovery.
- Capability to provide security through user authentication, user authorization, object access and object control.
- Distribution of processing elements in order to meet existing and future performance requirements.
- Ability to achieve application and component independence within a heterogeneous, distributed environment.
- Encapsulation of existing business logic for software reusability.
- Availability of support and development tools.

The following sections illustrate the approach and justification for selecting either a two-tiered or three-tiered client/server architecture used to implement the DCE prototype based on the factors previously described.

**2.3.1.1 Two-Tiered Client/Server Architecture.** A typical two-tiered client/server architecture incorporates a client tier and a server tier. The GUI resides on the client, the resource manager resides on the server, and the business logic can reside on either the client, the server, or both the client and the server. The primary interface to the resource manager is through the use of Structured Query Language (SQL) calls or stored procedures provided by the DBMS.

There are several limitations to two-tiered architectures including:

- Low levels of scalability, availability, and reliability for large or complex systems.
- Minimal security for distributed database updates using mechanisms such as ORACLE's SQL*NET and remote shell within the client configuration.
- Lack of portability for DBMS stored procedures across multiple platforms.
- Large demand on system resources to run applications using a significant amount of bandwidth for a large amount of network X-traffic and data transfers.
- No built in transaction management capabilities.

S&M, and other GCCS database applications, are implemented in a two-tiered architecture. The S&M application combines both the graphical user interface and business logic in the client tier. S&M's server tier uses the ORACLE DBMS as the RM. In the event that distributed RMs are accessed, ORACLE's SQL*NET is used to distribute data and provide consistent updates across multiple databases.

**2.3.1.2 Three-Tiered Client/Server Architecture.** A typical three-tiered client/server architecture incorporates separate tiers for the client, server, and resource manager. These three tiers provide separation and reduce coupling between the GUI, business logic, and resource manager processes, thus allowing components to be distributed throughout the system.

The general considerations for migration of the applications from a two-tiered to three-tiered client/server architecture expected to be demonstrated with the DCE prototype include:

- Increased performance, efficiency, and manageability of the distributed client server and resource manager application components.
- Increased scalability of the distributed components.
- Increased effective sharing of applications and data, thus promoting software reusability.

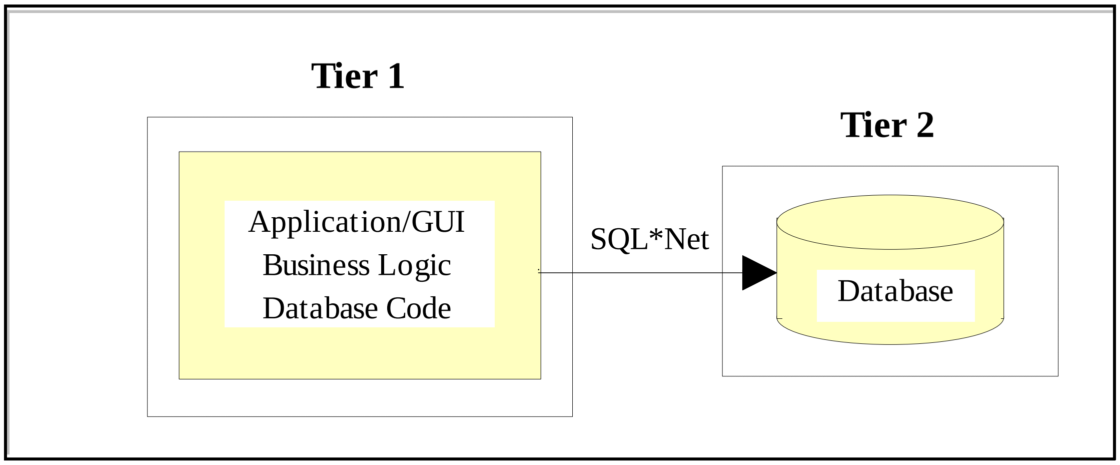
- Increased security and authentication for authorized use of servers and objects.
- Reduced development, maintenance, and administration costs for more complex systems.
- Management and distribution of transactions across multiple resource managers, thus, increasing transaction fault tolerance and performance.
- Open systems compliance through the use of a standardized interface infrastructure within the distributed processing environment.
- Allowing for versioning and multiple releases of clients and servers running concurrently for re-installations and upgrades while allowing the system to remain available for processing client requests.
- Reduced communication between client and server processes which can be minimized if network constraints require.
- Platform and environment independence through the use of platform independent data types in interfaces between components.

Disadvantages of a three-tiered architecture over an existing two-tiered solution include:

- Additional processing time required for server throughput, namely, lookup in a global namespace, security authentication and authorization of server objects, and administration.
- Additional system resources required to accommodate interface infrastructure and transaction management products.
- Difficulty debugging and testing server code during development. Difficulty in establishing initial testing environment. More complex administration required in all aspects of use.
- More configurable items to manage.
- Fewer support products and development tools.
- Requires more technical expertise.
- Products may be less mature; multi-vendor support needed.

For the DCE prototype, a three-tiered architecture was chosen to be implemented.

**2.3.1.3 Software Migration Strategy.** Developing a new three-tiered application and migrating an existing two-tiered application should result in the same end product. Therefore, a strategy is desired that can encompass the requirements of both scenarios. Below is a description of a general strategy for developing and migrating to three-tiered applications. Key factors in developing this strategy involve using the three-tiered model as a base for the strategy, attempting to provide a general target machine independent strategy with the ability to re-use code as appropriate, maximizing generated code, and minimizing changes to existing code in the case of application migration.



*Fi
gure 2-1: Existing Two-Tiered Database Application.*

An existing two-tiered GCCS database application as shown in Figure 2-1, consists of the application layer (GUI), business logic for processing business rules, and database I/O code, such as embedded SQL, to perform specified operations against the target database all combined into a single executable binary file as the first tier. Database operations typically access the database via standard or non-standard database interface calls and can be made directly or via a transport layer, such as SQL*NET for ORACLE. The RDBMS which processed these database calls and the database itself compose the second tier.

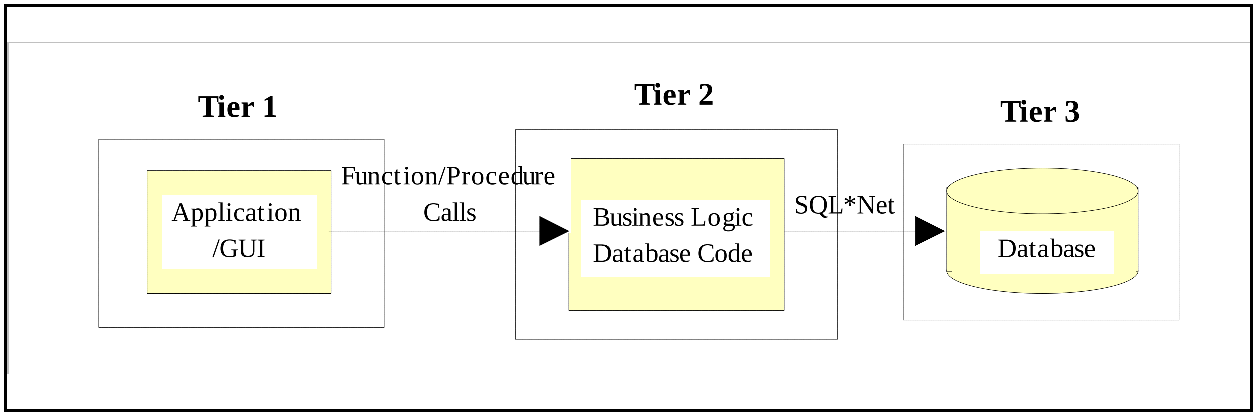


*Figure 2-2: Three-Tiered Database Application with Client/Server Interface.*

The first step in migrating an existing database application involves defining the interface between the business rules, consisting of business logic and database input/output (I/O) code, and the presentation layer (GUI) as shown in Figure 2-2. In truly modular code, this should be a fairly simple step as a well defined interface will already be defined between software modules.

This is the only step which should not be required in developing a new application, as a well defined interface between GUI and business logic is a key initial assumption when planning a three-tiered software design. As a result of this well defined interface, the code can be divided into two sections, a client which consists of the GUI, and a server which consists of business logic and database I/O code.

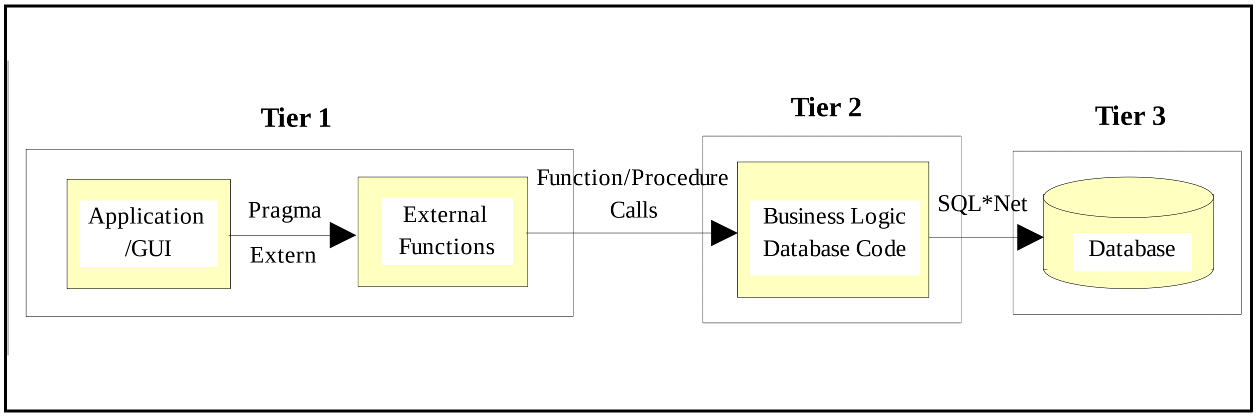


*Figure 2-3: Three-Tiered Database Application with Client External Routines.*

In the next step, instead of invoking the server code directly from the client via procedure and function calls, an external routine is now invoked from the client passing all required information to and from the server as shown in Figure 2-3. With this model, code in the external routine can easily interface with both the client code and the server code in an independent language (such as C). The primary reason for having external client routines will become apparent in the next migration step. Another advantage of this model is flexibility in the case where the client and server languages themselves are not compatible, but a third language is common between the two. For example, using Representational Specifications (RepSpecs) in Ada and pragma external calls to a C routine, an Ada client can use this model without specifically writing Ada bindings to the code. Also implied is reusability of routines with multiple GUI products without rewriting interface code because of the compatibility of the external routine.

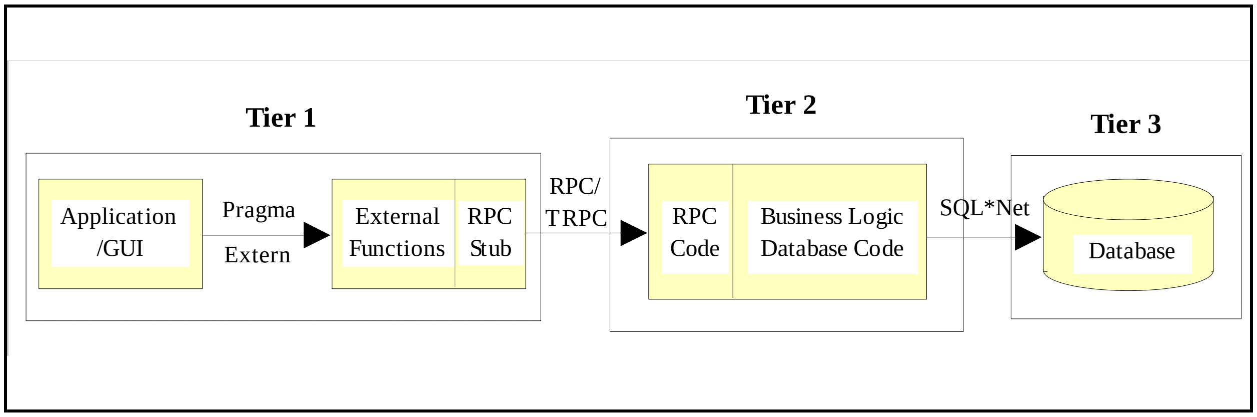


*Figure 2-4: Three-Tiered Database Application with Remote Procedure Calls.*

Using code generation from well defined Interface Definition Language (IDL) source files provides the ability to incorporate a platform independent interface between the client and server as shown in Figure 2-4. With DCE, this communication mechanism is implemented via Remote Procedure Call (RPC) stubs which are created with an IDL code generator. These stubs provide the means for the generation of RPCs on the client, and the reception of RPCs on the server to allow communication between clients and servers independent of machine boundaries. This strategy allows the code to be truly distributed as clients and servers now have the flexibility to be located on separate machines and on different hardware platforms. Also, since the language of the external routines is compatible with the RPCs, calls can be made directly to the stubs from the external routines in the client code independent of the client GUI.

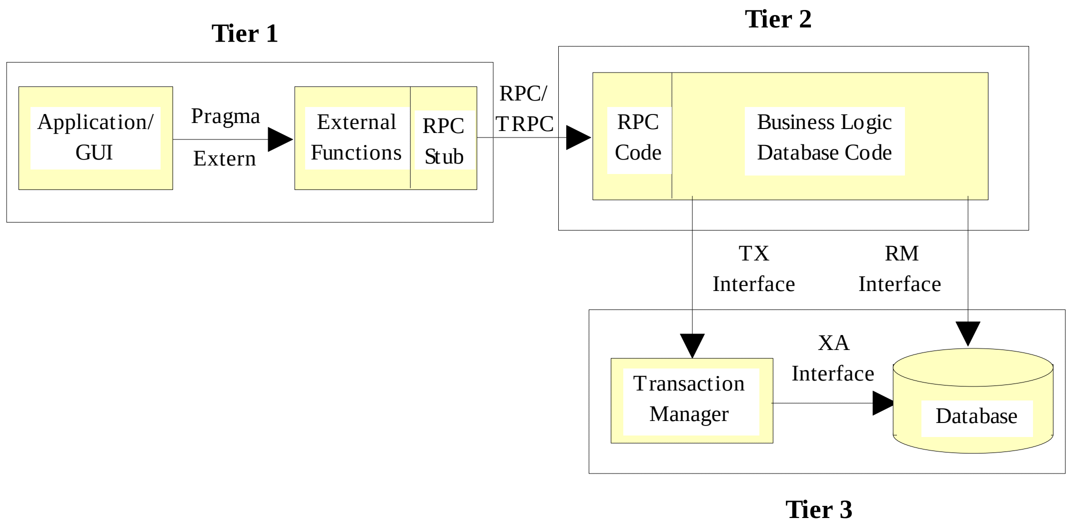


*Figure 2-5: Three-Tiered Database Application with Transaction Manager.*

In a traditional two-tiered database application, connection, disconnection, rollback, and commit commands are embedded in the application's database I/O code. The use of a transaction manager allows for the removal of all explicit commit, rollback, and database connection and disconnection code from the server by automatically providing transactional updates to the database as shown in Figure 2-5. The transaction manager interfaces with the database resource manager through a standard interface, such as the XA-Interface in ORACLE. The database resource manager (RM) interface should still be able to interpret database I/O code generated by the server and then convert those requests into well defined database calls to the specified database. The bounds of a transaction can then be defined on the server through the transaction (TX) interface such that if any database operation fails within these bounds, the database shall be automatically rolled back to its state before the beginning of the transaction. Only if all database operations succeed should the database be committed. This should remove all database specific information from the server which should now consist of database independent I/O code.

### 2.3.2 DCE Prototype Client/Server Architecture

The DCE prototype incorporates a three-tiered client/server architecture that uses DCE as the fundamental infrastructure to provide for distributed computing in a database application. The architecture of the DCE prototype comprises two overlapping architecture configurations. The first configuration, the DCE/Encina cell architecture, depicts the overall DCE and Encina monitor cell operational components. The second configuration, the software component design, demonstrates the logical configuration and interaction of the software application components. Both of these architectures provide the flexibility to support the various network topologies found in the GCCS and DII COEs.

**2.3.2.1 DCE/Encina Cell Architecture.** Although the DCE cell can be specified to any size and any configuration, the prototype delineates a single DCE cell architecture incorporating client and server nodes communicating across two local area network connected by a wide area network. This configuration was selected in order to demonstrate inter- and intra-operability of clients, servers, and resource managers across several host machines integrating multiple local area networks in a wide area network. The network topology provides the physical architecture that is overlayed by the DCE and Encina monitor cell architectures and software application architecture. The fundamental requirement for a single DCE cell architecture stipulates that clients and servers communicating primarily with each other be grouped into a single cell, thereby reducing intra-cell communication data traffic and processing overhead, and simplifying cell administration. Figure 2-6 illustrates the components within the DCE cell configuration. Since only a single DCE cell was configured, no intra-cell performance characteristics will be presented.

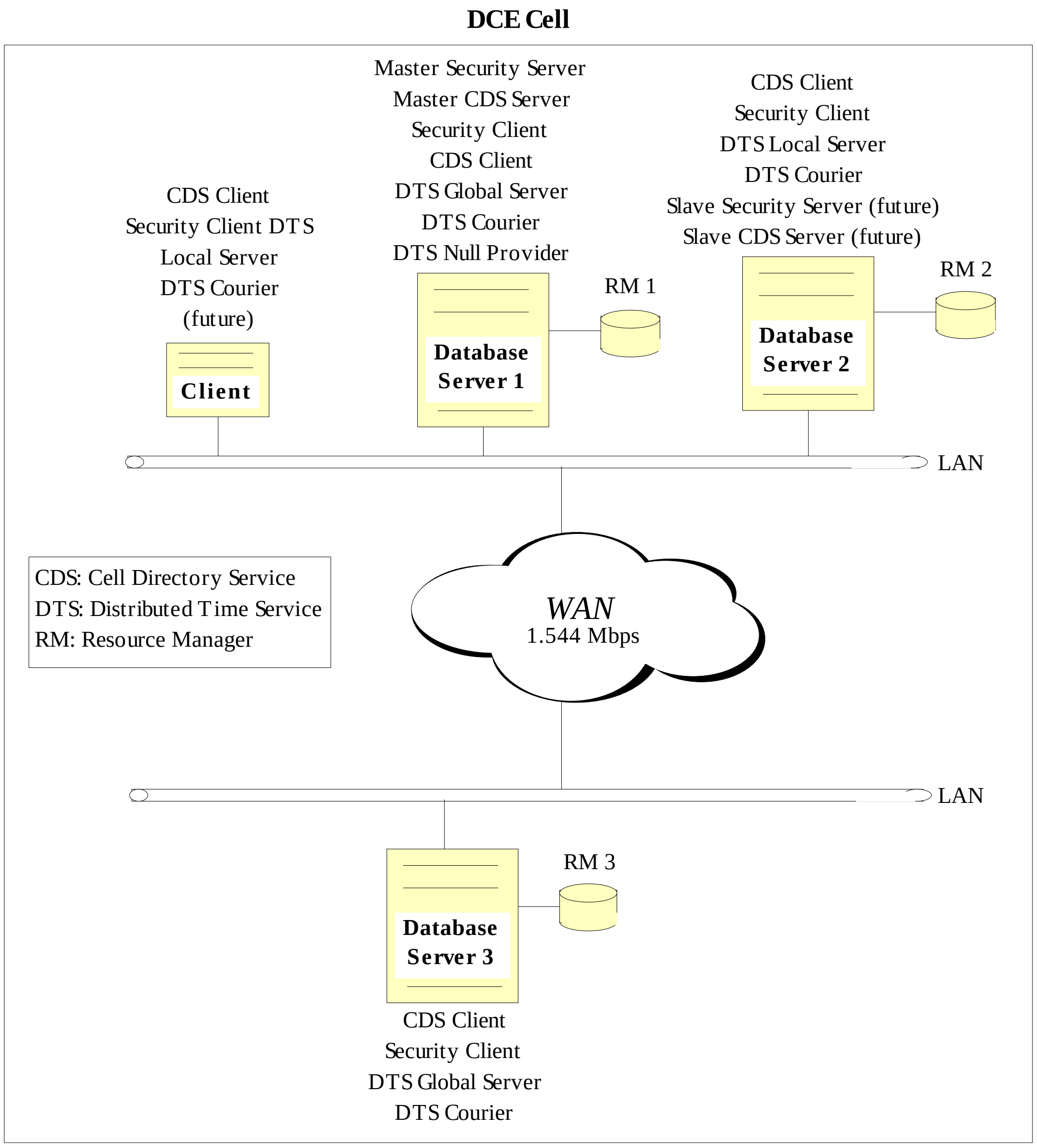


*Figure 2-6: DCE Cell Architecture.*

The DCE cell configuration incorporates both master and slave cell directory service (CDS) and security servers, global and local distributed time service (DTS) servers, and DCE clients. The master CDS server, master security server, CDS and security clients, DTS global time server, the DTS null time provider, and DTS courier are configured on a single server. CDS and security clients, the DTS local time server, and the DTS courier are configured on all alternate servers within the same LAN. Redundancy can be achieved through the configuration of slave CDS and security servers on an alternate server. The slave server maintains a backup to the master server and provides periodic updates of read-only replica databases containing CDS and security information that can be used to restore the master replicas in the event of system failure. CDS and security clients, DTS global time server, and a DTS courier are configured on an alternate server on a different LAN. Any additional alternate servers on the other LAN would be configured as CDS and security clients, DTS local time servers and DTS couriers, since one server on each LAN needs to contain these components of the DCE architecture.

All servers are configured either as DTS global or local time service providers. The DTS will request system times from all servers configured as time service providers. The information retrieved from the providers is used in calculating a new network wide time that will then be used to update all servers, thereby synchronizing all system times. A separate DTS global time server was configured for each LAN, and will be the primary time service provider for both LANs. Other servers located within a single LAN are configured as local time service providers. DTS couriers were configured instead of DTS clerks, since DTS clerks do not necessarily obtain system times from the DTS global time servers. DTS clerks only obtain system times from a required number of DTS local time servers as specified by a configurable system parameter. If the required number of DTS local time servers exceeds the actual number of DTS local time servers within a LAN, a time is then obtained from a DTS global time server. DTS global time servers provide system time updates across multiple local area networks, and therefore, are mandatory in synchronizing multiple networks.

Likewise, in configuring the prototype Encina monitor cell, the cell boundaries coincide with those of a single DCE cell. Although multiple Encina monitor cells can reside within a single DCE cell, the primary justification for the configuration of a single cell in the prototype is administration simplification of transaction manager components within the DCE cell.

The Encina monitor cell components overlay the DCE cell as illustrated in Figure 2-7. The shared file system (SFS) server, recoverable queuing service (RQS) server, cell manager (ecm), and node manager (enm) all reside on a single server that coincides with the primary DCE cell server. Since a node manager must reside on each machine that executes application servers, a node manager is configured for all alternate servers within the Encina monitor cell. However, it should be noted that if different versions of Encina are used within the context of a single DCE cell, multiple Encina monitor cells must be configured. Although basic CDS namespace and security information can be shared, version enhancements may preclude interchangeable functionality between versions.

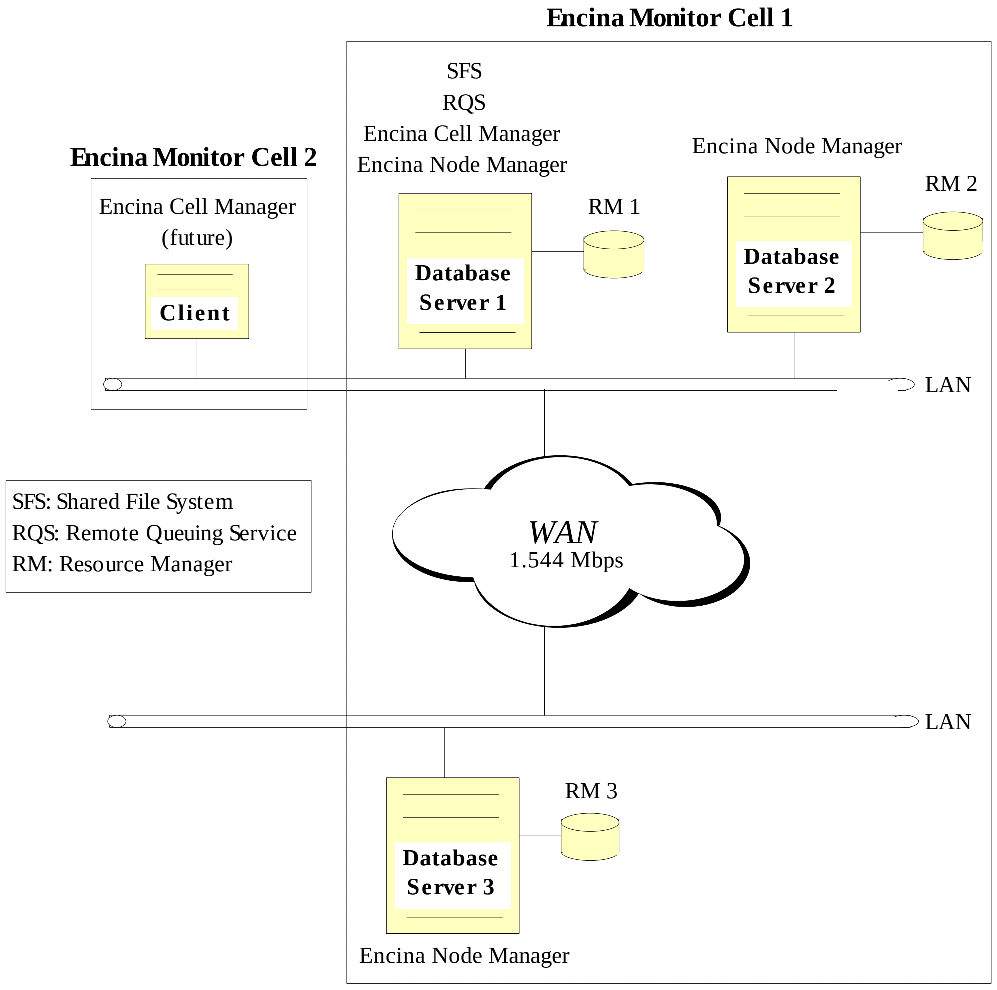


*Figure 2-7: Encina Cell Architecture.*

**<u>2.3.2.2 Application Software Overview.</u>** The S&M Add Carrier application server is implemented via an update server, as shown in Figure 2-8, using Encina TRPCs and the Encina transaction manager for insertions into multiple database tables as a single business function. By definition, TRPCs provide transactional updates so that if any portion of the Add Carrier update fails, all previous database updates are rolled back to an initial state before any updates occurred. The server was written in C and uses Pro*C embedded SQL to perform its database updates.

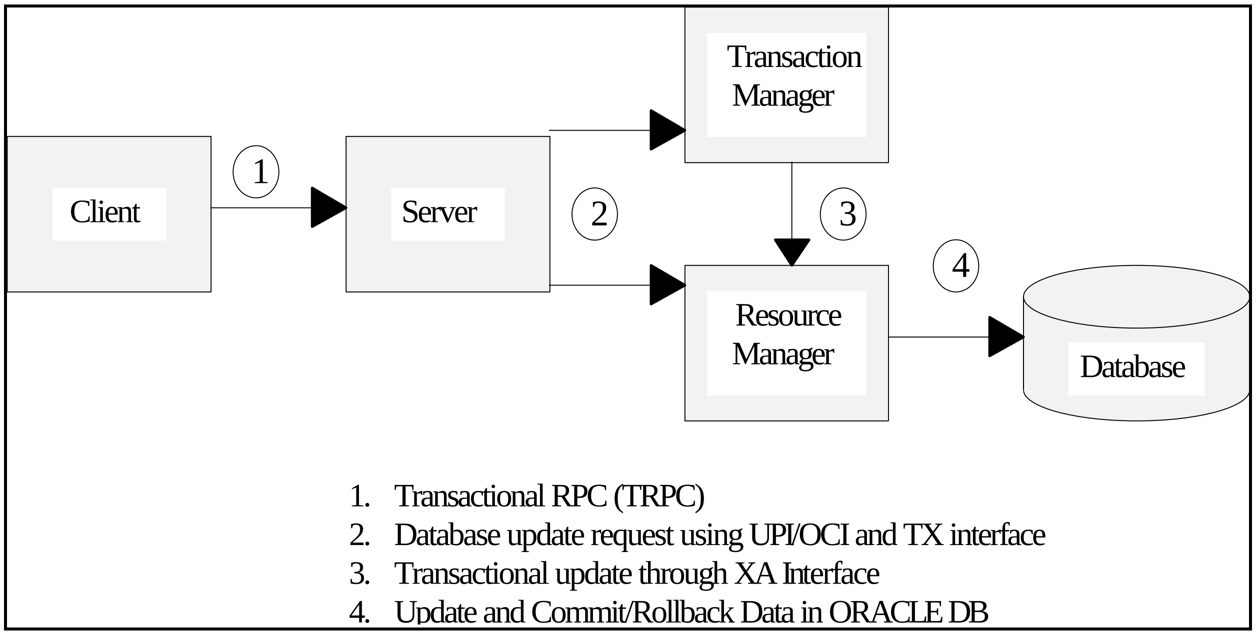
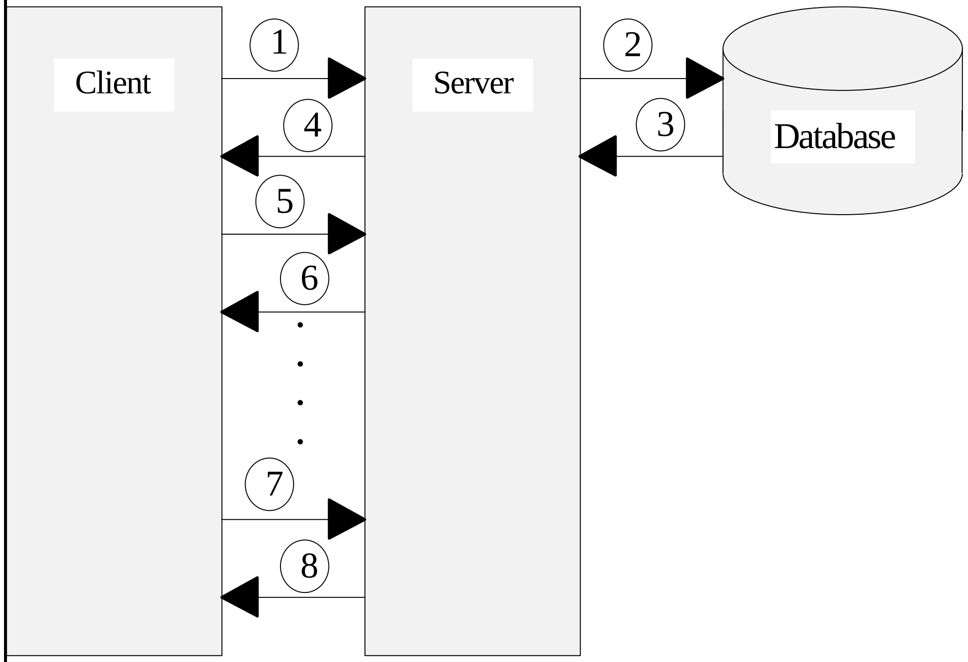

Transaction Manager
Client
1
Server
2
3
4
Resource Manager
Database
1. Transactional RPC (TRPC)
2. Database update request using UPI/OCI and TX interface
3. Transactional update through XA Interface
4. Update and Commit/Rollback Data in ORACLE DB

*Figure 2-8: Update Server Architecture.*

The S&M Query GEOLOCs Lookup query server, as shown in Figure 2-9, is implemented using DCE RPCs and the Encina resource manager for querying from the outer join of two database tables. The initial RPC passes the SQL statement to the server from the client and returns back the first page of information as well as an RPC context handle containing resource information from the server. Additional RPCs are then performed as needed to retrieve records desired by user paging requests using this context handle. Paging provides the ability to limit the data passed between the client and server to only what is displayed on the client. A final RPC is executed to free the context handle and other resources on the server upon termination of the query. It was written in C and uses Pro*C embedded SQL to perform its database query.

The ODS query server was implemented in a similar fashion as the S&M Query GEOLOCs Lookup server described above.



1. Initialize query RPC
2. Database query to ORACLE DB
3. Database query results returned from ORACLE DB
4. Initialize RPC results -> 1st page, context handle, server still maintains query results
5. Get Page RPC
6. Get Page RPC results -> desired page returned
7. Terminate Query RPC
8. Null context handle returned, resources freed on server

*Figure 2-9: Query Server Architecture.*

All application servers are replicated on each database server machine as each application server is restricted to communicating with a single RM. Also, each application server is configured as an Encina Monitor Daemon (mond), with three Processing Agents (PAs). Essentially three versions of each server are running to support each database. This was necessary due to the current lack of support for multi-threaded applications by the ORACLE XA Interface, otherwise a single multi-threaded PA with support for three threads could also support three requests simultaneously. Figure 2-10 displays the implementation of this strategy.

The GUI client was written in Tk/Tcl with supporting routines written in C (see more information in Paragraph 2.4.4.2). An exact copy of each client is located on each client machine for generating RPC/TRPC requests to allow the application to be executed locally on each machine.

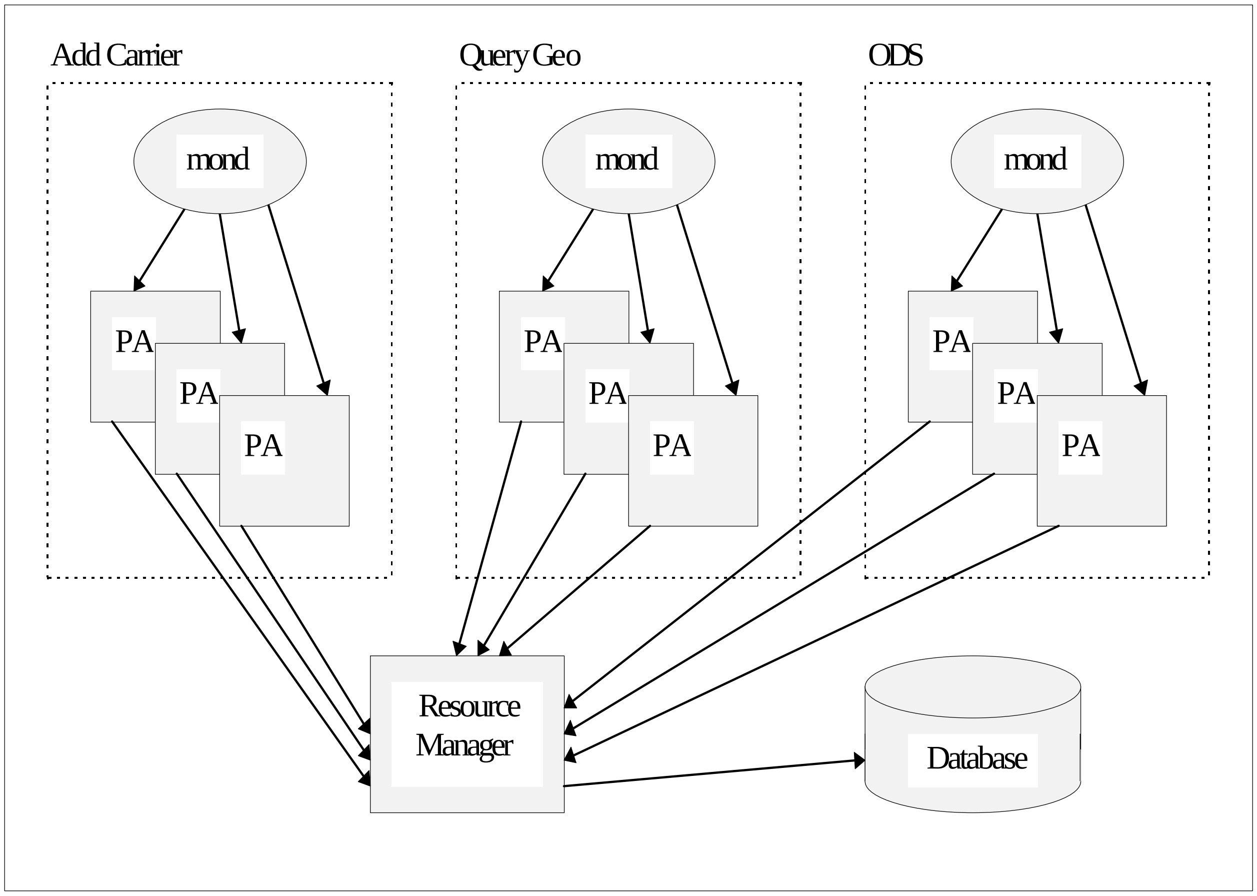


*Figure 2-10: Application Server Implementation.*

## 2.3.3 DCE Prototype Operating Environment

The DCE prototype operating environment includes the operational environment, hardware environment, and software environment necessary to implement and deploy the DCE prototype S&M application servers, DCE, and Encina products.

**2.3.3.1 Operational Environment for Scheduling and Movement.** The S&M application currently requires a minimal operating environment as shown in Table 2-2.

*Table 2-2: Scheduling and Movement (S&M) Minimal Operating Environment.*

| Requirement | Product |
|---|---|
| RAM Required | 24 MB |
| Operating System | Solaris 2.3 |
| RDBMS | Oracle V7 |
| Other Req'd GCCS Segments | GCCS Core Database |
| GUI | Screen Machine V 1.4.1 |
| Disk Usage | .35 MB Client Segment<br>210 MB Server Segment |

The client installation is simply an 'rsh' command which remotely launches the application on another server, but sends X-Windows display information back to the client.

**2.3.3.2 Operational Environment for DCE/Encina.** DCE 1.0.3a and Encina 1.1 require a minimal operating environment as shown in Table 2-3.

*Table 2-3: Transarc DCE 1.0.3 and Encina 1.1 Minimal Operating Environment.*

| Component | Operating System | RAM Required | SWAP Space | Disk Usage |
|---|---|---|---|---|
| DCE 1.0.3a Client | Solaris 2.3 | 24 MB | 48 MB | 20 MB + Appl. Client Binary |
| DCE 1.0.3a Server (w/ Security, CDS, etc.) | Solaris 2.3 | 32 MB | 80 MB | 57 MB + Appl. Server Binary |
| Encina 1.1 | Solaris 2.3 | 32 MB | 34 MB + 8 MB per server + 4 MB per additional PA | 67.5 MB + Appl. Server Binary |

**2.3.3.3 Software Development Environment.** The Software Development environment is composed of several components as shown in Table 2-4.

*Table 2-4: DCE Prototype Software Development Environment.*

| Component | Product |
|---|---|
| Operating System | Solaris V2.3 |
| Compilers | VADS V2.1e (Ada)<br>SPARCCompiler C V2.0.1 |
| GUIs | Tk V4.0<br>Tcl V7.4<br>Screen Machine V1.4.1 |
| Database | Oracle V7.1.3<br>Oracle V7.1.4.1 |
| DCE | Transarc DCE V1.0.3a |
| Transaction Mgmt System | Encina V1.1 |

## 2.4 PRODUCT AND COMPONENT ANALYSIS

This section provides product considerations with respect to fulfilling functional and non-functional requirements, product maturity, and vendor product support of the DCE prototype. The

DCE and Encina overview and analysis section provide a description of the DCE and Encina products with respect to the DCE prototype functional and non-functional requirements. GUI products are also analyzed in order to propose a complete migration strategy for both existing and new database applications.

### 2.4.1 DCE and Encina Overview and Analysis

This section provides a summary of DCE’s and Encina’s abilities to satisfy the functional and non-functional requirements outlined in Paragraph 2.2. Two different evaluations are made in this section. The first evaluation addresses DCE’s ability to solely satisfy the requirements, while the second evaluation describes DCE and Encina’s combined ability to satisfy the requirements. Both evaluations are based primarily on information uncovered during the prototyping effort and secondarily on information gathered during research.

For each requirement area, the evaluation is indicated in one of four ways. A ✓ indicates that the product satisfies the requirement. A ✓- indicates that the product partially satisfies the requirement. A ✓+ indicates that the product more than satisfies the requirement or provides enhancements. An ✗ indicates that the product does not satisfy the requirement. A ★ indicates that SRA was able to prototype a particular requirement area with the specified product in SRA’s DCE prototype.

#### <u>2.4.1.1 Distributed Resource Management.</u>

| | |
|---|---|
| DCE | ✗ |
| DCE with Encina ★ | ✓+ |

DCE does not provide any functionality in the area of distributed resource management, but does provide the framework for being able to develop a resource management system on top of it. Encina contains a distributed resource management framework which allows applications to access RMs by name from within application servers. This is true whether or not the RM resides on the same server as the application. In the DCE prototype, ODS allows users to connect seamlessly to local or remote databases by sending RPC/TRPC requests from the client to an application server which is connected to the desired resource manager.

#### <u>2.4.1.2 Transaction Management.</u>

| | |
|---|---|
| DCE | ✗ |
| DCE with Encina ★ | ✓ |

DCE in conjunction with Encina's tpm (Transaction Processing Monitor) and ORACLE's XA interface facilities is able to perform database updates, deletes and insertions by grouping multiple operations into a single business rule or transaction. If any portion of the database operation fails, all database changes from the beginning of the transaction are rolled back. In addition, the application server contains no explicit database connection, disconnection, rollback, or commit statements as these are all handled by the transaction manager. DCE alone provides no built in transactional facilities.

## **2.4.1.3 Data Retrieval.**

| | |
|---|---|
| DCE | ✓- |
| DCE with Encina ★ | ✓ |

DCE RPCs provide the notion of a context handle which is defined by the initial RPC generated by the client and processed by the application server. The context handle can then be passed back and forth between the client and application server on subsequent RPC calls. The final RPC from the client should clear this context handle and free any allocated server resources. Additionally, any loss of communication between the client and application server will result in a routine called a context_rundown executing on the server to free the context handle and other needed resources. This strategy implements the required functionality for creating a query server as per the functional requirements. After the query is performed, the data is stored in memory on the application server, and a pointer to that memory is used as the context handle. Demand paging is then available between the client and application server as needed. Other than providing a method for accessing distributed resource managers as mentioned in Paragraph 2.4.1.1, Encina does not provide any additional functionality to further satisfy this requirement.

## **2.4.1.4 GCCS Current Database Permissions Schema.**

| | |
|---|---|
| DCE | ✗ |
| DCE with Encina ★ | ✓+ |

DCE cannot solely fulfill any database requirements due to its lack of a built in distributed resource manager as mentioned in Paragraph 2.4.1.1. Encina, however, is able to implement the GCCS 2.0 Database Permissions schema within an application programmatically (i.e., routines must be written in application servers to access permissions tables and verify permissions for the users’ IDs and access permissions). Additionally, using DCE’s access control list (ACL) facility can provides for a more secure implementation of the current GCCS permissions scheme by determining what user can implement what business functions.

## **2.4.1.5 GCCS Future Database Permissions Schema.**

| | |
|---|---|
| DCE | ✗ |
| DCE with Encina | ✗ |

Encina is not able to implement a future GCCS permissions schema which is based on database views to verify a user's ability to access data. This is due to limitations in the XA-Interface to ORACLE which requires a single login from the resource manager via a connection string. Therefore, the application server is unable to connect to the database as a specific user and consequently is unable to access the views correctly. A concept called "delegation" should deal with this known limitation in future ORACLE and Encina releases and will allow the application server to assume a user role when connecting to the database. When this functionality is added, this requirement should be satisfied.

### 2.4.1.6 Cross-Platform Independence.

| | |
|---|---|
| DCE | ✓ |
| DCE with Encina | ✓ |

DCE provides cross-platform independence through the use of data marshalling and machine independent Interface Definition Language (IDL) data types. Clients and servers can run on different platforms as long as DCE has been ported to those platforms. Encina additionally provides this ability as well, but has been ported to a subset of the DCE platforms. The DCE prototype, however, was implemented on a single platform testbed of Solaris 2.3 SPARCStations, and this requirement was not implemented in the DCE prototype.

### 2.4.1.7 Product Interoperability.

| | |
|---|---|
| DCE | ✓ |
| DCE with Encina | ✓ |

The Open Software Foundation (OSF) has verified Transarc DCE's interoperability with products based on open systems standards. Transarc DCE uses standard APIs and is compatible with products designed for other vendors’ implementations of DCE. The DCE prototype verified interoperability with Encina 1.1, but no other products were tested. There are, however, a number of DCE compatible products on the market that, with the OSF guarantee, should be compatible with Transarc DCE. Additionally, add-on products for Encina should also become available, although none were tested for this prototype.

### 2.4.1.8 Usability and Administration.

| | |
|---|---|
| DCE ★ | ✓- |
| DCE with Encina ★ | ✓- |

DCE is difficult to administer especially with its script based administration packages such as cdscp and rpccp. In the event of a failure, manually developed scripts provide the only method for restarting the applications and DCE servers without administrator intervention. Additionally, all logging and error handling for application servers must be performed programmatically. Encina provides additional administration tools which help to increase the usability of DCE including the monitor daemon (mond) process which controls server processing agents (PAs), will log errors, and automatically initializes, restarts, and shuts down servers without the need of user-developed scripts. However, administration of the DCE cell, on which Encina is dependent, is not significantly improved by these value add-ons of Encina. Value added products to aid in DCE administration definitely should be considered for a fielded system and are also available in more recent versions of Transarc's DCE and Encina.

### 2.4.1.9 Security

| | |
|---|---|
| DCE | ✓- |
| DCE with Encina | ✓- |

DCE uses several different mechanisms to authenticate users and to verify user authorization levels. These mechanisms include the DCE registry service, authentication service, privilege service, access control list facility and login facility. The privilege service and access control facility use the DCE security database to verify user credentials and privileges, thus controlling access to resources. Additionally, the DCE login facility allows a user's UNIX login access to be mirrored within DCE. This aspect, however, must eventually be integrated with the UNIX login as per user requirements for a single login mechanism. Encina uses DCE security, but does not further satisfy the requirements.

### 2.4.1.10 Performance

| | |
|---|---|
| DCE | ✗ |
| DCE with Encina ★ | ✓+ |

It is not possible to evaluate DCE's performance characteristics alone because the desired functionality of the DCE prototype could not have been achieved without Encina. The performance tests (see Paragraph 2.5) conducted by SRA demonstrate that the DCE prototype of the S&M application performs as well and in most cases performs better than the original S&M application. It is apparent that overhead introduced by such aspects of DCE as security lookup did not significantly hinder DCE's performance.

### 2.4.1.11 Scalability.

| | |
|---|---|
| DCE ★ | ✓ |
| DCE with Encina ★ | ✓ |

DCE provides the ability to distribute processing into client and server segments, thus demonstrating application scalability across multiple machines. Encina, adds onto this functionality by providing load balancing and transparent bindings to allow processing to be automatically distributed, although this functionality was not implemented in the DCE prototype. With ODS, the user makes the decision of where to process the data, and thereby provides a scalable system.

### 2.4.1.12 Fault Tolerance.

| | |
|---|---|
| DCE | ✗ |
| DCE with Encina ★ | ✓ |

DCE incorporates a number of processes or daemons to provide the directory, security, distributed time services, and cell managers. These processes or daemons may drop out, causing DCE system failures. If DCE security or cell directory service redundancy is configured, some failures can be overcome without degradation to the system. Otherwise, DCE does not explicitly provide fault tolerance Encina's use of the mond process with multiple PAs as previously described does provide high server availability. Encina also provides reliability by using the transaction monitor capabilities to guarantee that all transactional database updates either completely succeed or fail and rollback the database. Maintainability is provided by version control of the Encina TRPC and DCE RPC interfaces which allows servers to concurrently support multiple versions of clients and allows for ease of configuration management when installing and/or upgrading software. This strategy also allows continued system availability to users during maintenance activities.

### **2.4.1.13 Existing GCCS Database Application Implementations.**

| | |
|---|---|
| DCE | ✓- |
| DCE with Encina | ✓- |

Both DCE and Encina were incompatible with ScreenMachine and Gain Momentum GUI technologies due to specific limitations of these products (see Paragraph 2.4.2). However, with Tk/Tcl, the DCE prototype was able to be implemented and should be compatible with any other POSIX compliant threaded GUI.

### **2.4.1.14 Application Specific Requirements.**

| | |
|---|---|
| DCE | ✓- |
| DCE with Encina ★ | ✓ |

DCE and Encina together provide features for the implementation of the GEOLOC lookup functionality. In the DCE prototype, the Encina RM was used to access the database, while the DCE RPCs using context handles were used to transmit data from the server to the client. ODS was implemented in a similar manner. The Add Carrier business rule functionality was completely implemented via Encina TRPCs. In short, all requirements for add carrier, GEOLOC lookup and ODS were satisfied by the DCE prototype.

## 2.4.2 Graphical User Interface (GUI) Products

The initial S&M Add Carrier application was written in Ada and uses Objective Interface Systems' (OIS) ScreenMachine product as a GUI interface. ScreenMachine is an Ada source code generator which allows the user to paint screens and then generates modifiable Ada source code which can be coupled with developed Ada code for database applications.

The initial strategy was to insert 'pragma Interface' commands into the Ada Code to call the appropriate C routines which would perform the DCE/Encina RPC/TRPC calls. This method required RepSpecs of all Ada records in order to guarantee machine representation of the data in C.

**2.4.2.1 ScreenMachine.** Although the Ada code compiled, problems were encountered upon issuance of the mon_InitClient Encina call from the client. The problem was determined to be related to the fact that ScreenMachine uses Ada tasking to implement its GUI. These tasks are then compiled by the Verdix Ada Development System (VADS) compiler V2.1e into non POSIX compliant threads which are not compatible with the DCE/Encina RPCs/TRPCs being issued by the client stubs. This problem manifested itself by causing the GUI to hang indefinitely after generating field exits which should have immediately been processed (deadlocked waiting for system resources).

Therefore, an impasse was reached. ScreenMachine (OIS) and VADS (Rational Software Corporation) suggested possible future solutions using different versions of compilers that might alleviate the problem by implementing the Ada tasking in a way which was thread-safe and POSIX compliant. OIS suggested the GNAT Ada '95 compiler as a solution, and were already working on porting their product to this platform in the near future. Rational suggested a multi-processor version of their compiler called VADS_MP which would theoretically also work, but OIS was not planning a port to this compiler.

**2.4.2.2 Tk/Tcl.** Although Tk 4.0(Tcl's X window *Toolkit*)/Tcl 7.4(*Tool Command Language*) does not claim to be thread-safe, the GUI built using this tool worked well with the DCE/Encina environment. The alpha release of the Tk/Tcl software supposedly specifically addresses the issue of threads, but it has not been fully tested yet; therefore, it is not recommended to be used in a formal software development environment.

Tk/Tcl is a script language similar to UNIX shell languages that has enough programmability to build complex applications using the scripts alone. Most of the client level code in the DCE prototype that manipulate the GUI for the Add Carrier application are written in Tk/Tcl. In addition to built-in Tk/Tcl procedures, Tk/Tcl provides capability to write application-specific procedures using Tk/Tcl's C library. The procedures that interact with DCE/Encina RPC/TRPC are written as C functions using these libraries. These functions are incorporated into the Tk/Tcl script language by recompiling (using SPARC C compiler V2.0) the Tk/Tcl interpreter, thus the new functions become part of the Tk/Tcl interpreter's built-in procedure. Tk/Tcl is flexible enough so that it can easily merge C codes with Tk/Tcl codes, and it also provides extensive C library that can be used to build customize widgets. There are many user groups that are writing extensions and tools for Tk/Tcl such as GUI builders, and these tools should facilitate and enhance future developments efforts.

**2.4.2.3 PowerBuilder and Other Alternative GUIs.** Powersoft's PowerBuilder 4.0 is the first release of the product that provides a mechanism for developing two-tiered client/server applications and graphical user interfaces operating on the SUN hardware platform (Solaris 2.4). PowerBuilder supports a heterogeneous, distributed computing environment by providing server independence through direct connectivity between various types of server databases, including Oracle, Sybase, Microsoft SQL server, and also supports databases incorporating the ODBC standard. Likewise, PowerBuilder 4.0 provides the ability to interface with existing C libraries and incorporates native object libraries that are compatible across multiple platforms.

PowerBuilder 4.0 was not used in the DCE prototype development primarily because it supports only a two-tiered client/server architecture and thus, lacks DCE (and CORBA) support. Likewise, it does not provide thread-safe libraries. Future releases according to Powersoft could be DCE compliant.

Transarc also has provided a list of compatible GUIs already proven to work with its product for SUN platforms under the Solaris operating system. Appendix C contains a listing of these products.

## 2.5 PERFORMANCE TESTING OF DCE PROTOTYPE

The following sections detail the test environment, test cases, and test results pertaining to performance tests. These tests compare various database operations using the DCE prototype versus using the original S&M application executing the same database operations. The original S&M application can use either the X or SQL*NET protocol to display the results of remote database operations and the performance results from both protocols will be compared to the results from the DCE prototype.

### 2.5.1 Test Environment

The tests were conducted in a three server environment. Two servers are connected via a LAN using a 10 megabits per second (Mbps) Ethernet link. These two servers are SUN SPARCstation 2000's, named JDIC1 and JDIC3. The third server in the test environment, a SPARCstation 1000 named JDIC4, connects to JDIC1 and JDIC3 by a simulated WAN link. For specific details on each server's specifications, refer to Table 2-5.

*Table 2-5: Server Specifications.*

| Server Name | Type of Server | RAM | Number of CPUs | Swap Space |
|---|---|---|---|---|
| JDIC1 | SPARCstation 2000 | 800 Mb | 4@40 MHZ each | 800 Mb |
| JDIC3 | SPARCstation 2000 | 640 Mb | 4@40 MHZ each | 1.25 Gb |
| JDIC4 | SPARCstation 1000 | 512 Mb | 4@40 MHZ each | 825 Mb |

To simulate a WAN environment, a CISCO 4000 router is attached to the LAN containing JDIC1 and JDIC3. Two channel service units/data service units (CSUs/DSUs) are connected to the router in succession. The other end of the CSUs/DSUs are connected to the LAN containing JDIC4 via another CISCO 4000 router, thus completing the WAN link. The purpose of the routers is to simulate the WAN link. However, in order to vary speed of the WAN link, a parameter in the testing, the two CSUs/DSUs are needed. These devices allow for WAN link capacity to be varied incrementally from 64 kbps to 1.54 Mbps (T1) speeds. Figure 2-11 further illustrates the test environment.

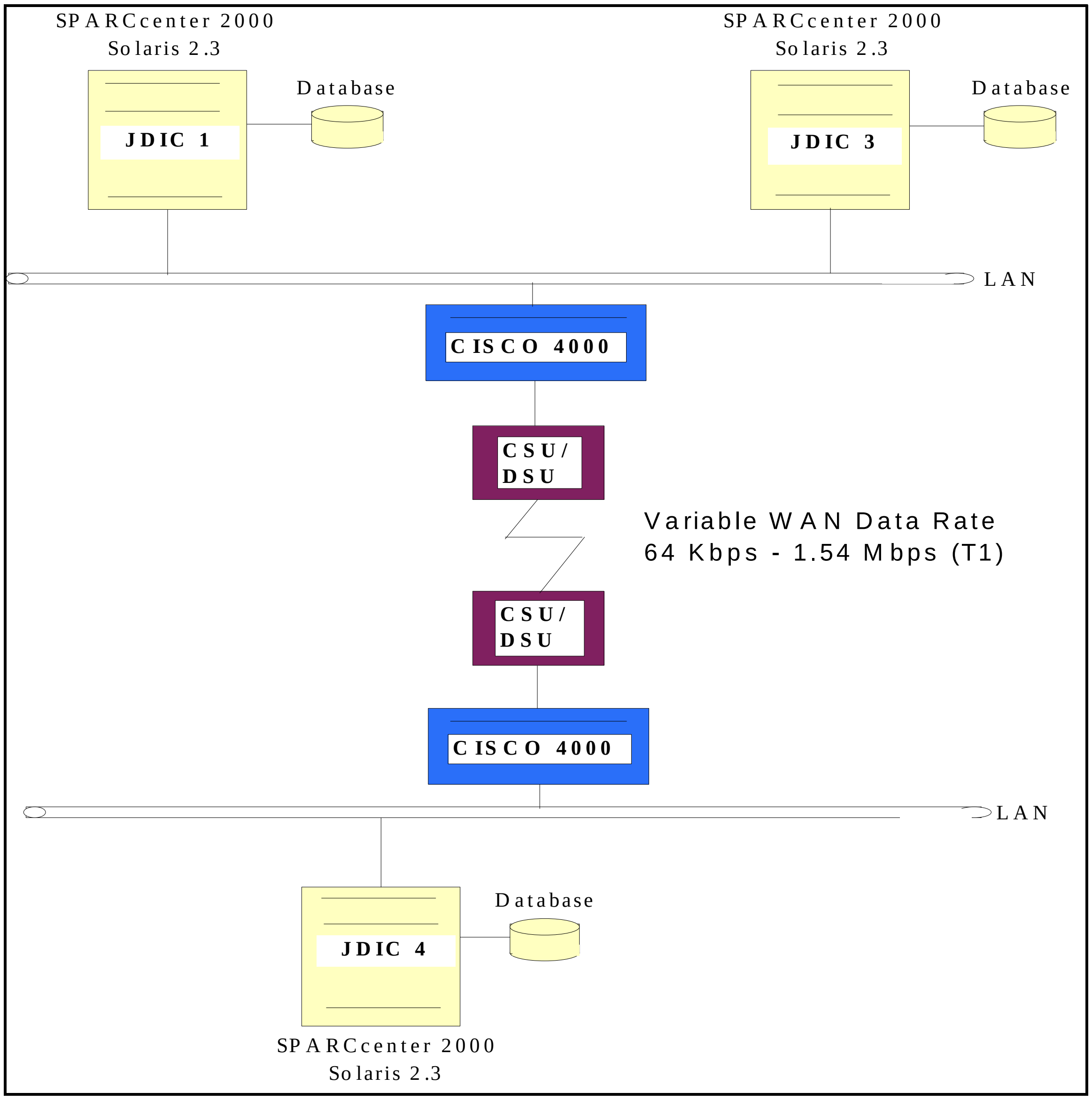


*Figure 2-11: The Performance Testing Environment.*

### 2.5.2 Test Cases

Several different tests were conducted to compare the performance of the three-tiered DCE prototype to the two-tiered original S&M application using both the X and SQL*NET communication protocols. The X protocol provides the ability to have a client process accessing a database on a remote server while displaying the results on the local server. SQL*NET allows a local client application to access a remote database and display the results locally. The tests measured the amount of time for the application to perform an operation on a database and return the results of the operation to the user's screen. Three different database operations are initiated for each test case. These database operations include retrieving all GEOLOCS (approximately 52,000 rows of data) from the database, retrieving all GEOLOCS in Germany (approximately 2,800 rows of data) from the database, and adding an air carrier to the database. These operations are typical actions performed by S&M users. Identical transactions were used in the testing to help ensure that any differences in performance result from differences in software architecture and communication protocols rather than the test cases themselves.

The first two test cases measure the time the applications take to query a local database using both the DCE prototype and the original S&M application. These two cases show only the differences between the DCE prototype and the original S&M application. However, DCE, X and SQL*NET are compared twice for each type of database operation using JDIC1 first as the display and then JDIC4 as the display, the only common attribute between DCE, X and SQL*NET in the test cases. Therefore, although the display is not the main criterion in the tests, it serves as a good reference for the rest of this section when describing each test case.

For example, the All GEOLOCS query for DCE displayed at JDIC1, client at JDIC1 and database at JDIC4 is referenced as the DCE All GEOLOCS (Display JDIC1). The other variables, the server on which the client resides and the database being accessed, change depending on the nature of the tests. When the display is set to JDIC1, SQL*NET and DCE tests use JDIC1 as the client for the application while accessing JDIC3's database for the LAN test and JDIC4's database for WAN tests. X tests have the display set to JDIC1, but the client and database both reside on JDIC3 for the LAN test and the client and database both reside on JDIC4 for the WAN tests. The results in the simulated WAN environment are collected at the data rates of 1.544 Mbps (T1), 512 Kbps, 256 Kbps, 128 Kbps, and 64 Kbps to determine the effects on performance due to lower link capacities. For the test cases with JDIC4 as the display, the DCE and SQL*NET tests are initiated with JDIC4 as the client and JDIC3 as the database. The X tests have the client and database on JDIC3, with only the display set to JDIC4. No tests in the LAN environment can be conducted with JDIC4 as the display because no other server resides on JDIC4's LAN. Refer to Table 2-6 for further illustration of all the test cases.

*Table 2-6: Summary of Test Cases.*

| Test Cases | Server Client Resides | Server Containing Display | Database Server | Link Capacity |
|---|---|---|---|---|
| Local DCE case | JDIC1 | JDIC1 | JDIC1 | Not Applicable |
| Local non-DCE case | JDIC1 | JDIC1 | JDIC1 | Not Applicable |
| DCE LAN case | JDIC1 | JDIC1 | JDIC3 | 10 Mbps |
| SQL*NET LAN case | JDIC1 | JDIC1 | JDIC3 | 10 Mbps |
| X LAN case | JDIC3 | JDIC1 | JDIC3 | 10 Mbps |
| DCE WAN cases (DISPLAY JDIC1) | JDIC1 | JDIC1 | JDIC4 | 64, 128, 256, 512, 1540 Kbps |

| Test Cases | Server Client Resides | Server Containing Display | Database Server | Link Capacity |
|---|---|---|---|---|
| SQL WAN cases (DISPLAY JDIC1) | JDIC1 | JDIC1 | JDIC4 | 64, 128, 256, 512, 1540 Kbps |
| X WAN case (DISPLAY JDIC1) | JDIC4 | JDIC1 | JDIC4 | 64, 128, 256, 512, 1540 Kbps |
| DCE WAN cases (DISPLAY JDIC4) | JDIC4 | JDIC4 | JDIC3 | 64, 128, 256, 512, 1540 Kbps |
| SQL WAN cases (DISPLAY JDIC4) | JDIC4 | JDIC4 | JDIC3 | 64, 128, 256, 512, 1540 Kbps |
| X WAN cases (DISPLAY JDIC4) | JDIC3 | JDIC4 | JDIC3 | 64, 128, 256, 512, 1540 Kbps |

### 2.5.3 Test Results

The following graphs illustrate the results of the test cases described in the previous section. Each graph contains the results for the DCE, X and SQL*NET test cases for each of the database operation and for each display setting. The raw numerical data can be found in Appendix D.

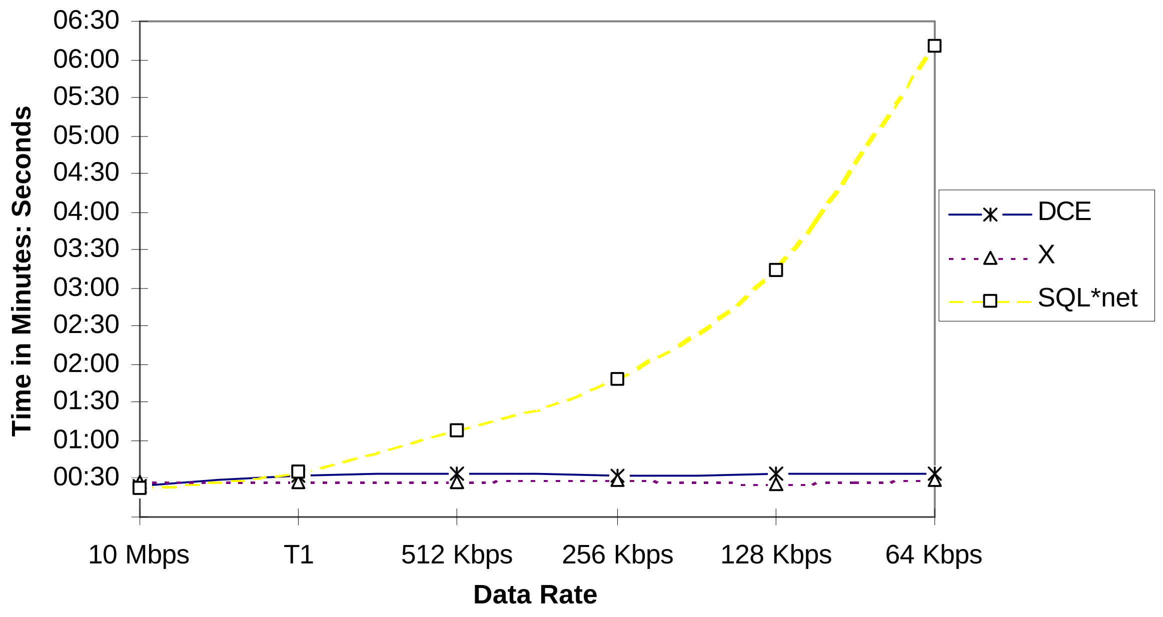


*Figure 2-12: Comparison of DCE, X, and SQL*NET for ALL GEOLOCS Query (Display JDIC1).*

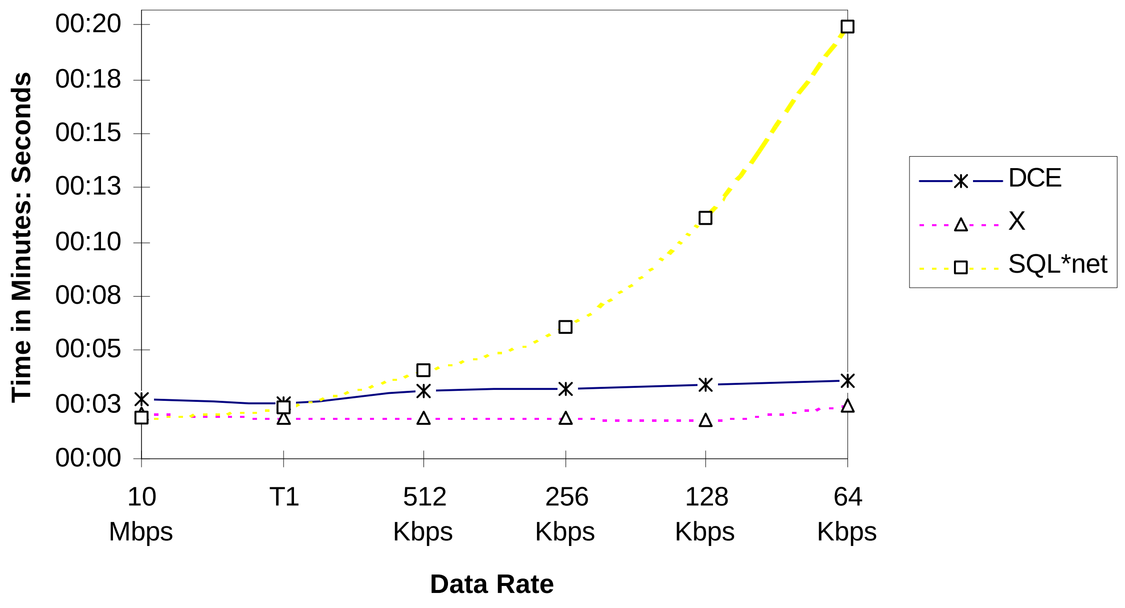


*Figure 2-13: Comparison of DCE, X and SQL*NET for GEOLOCS in Germany Query (Display JDIC1).*

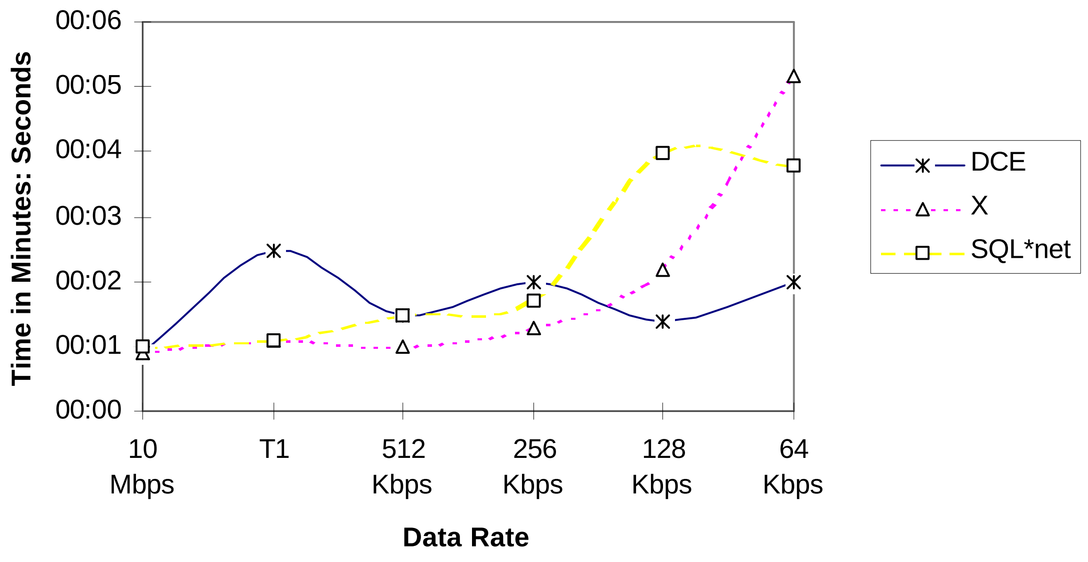


*Figure 2-14: Comparison of DCE, X and SQL*NET for Addition of an Air Carrier (Display JDIC1).*

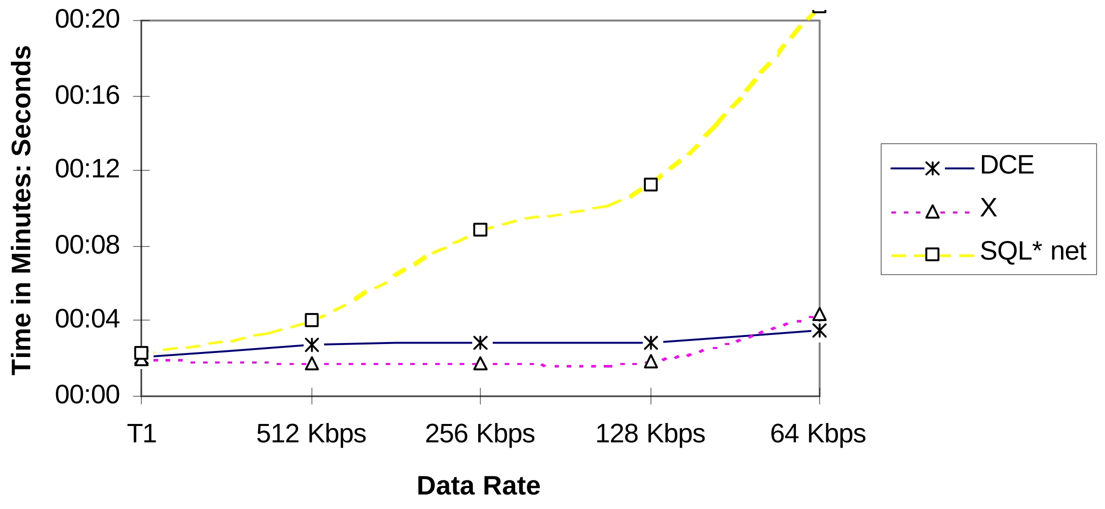


*Figure 2-15: Comparison of DCE, X and SQL*NET for All GEOLOCS Query (Display JDIC4).*

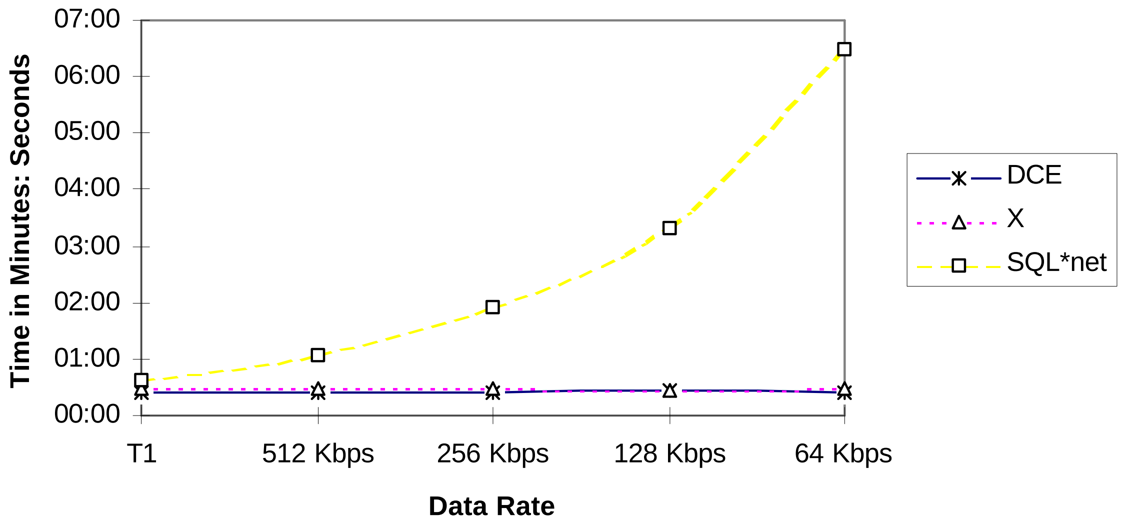


*Figure 2-16: Comparison of DCE, X and SQL*NET for Query of GEOLOCS in Germany (Display JDIC4).*

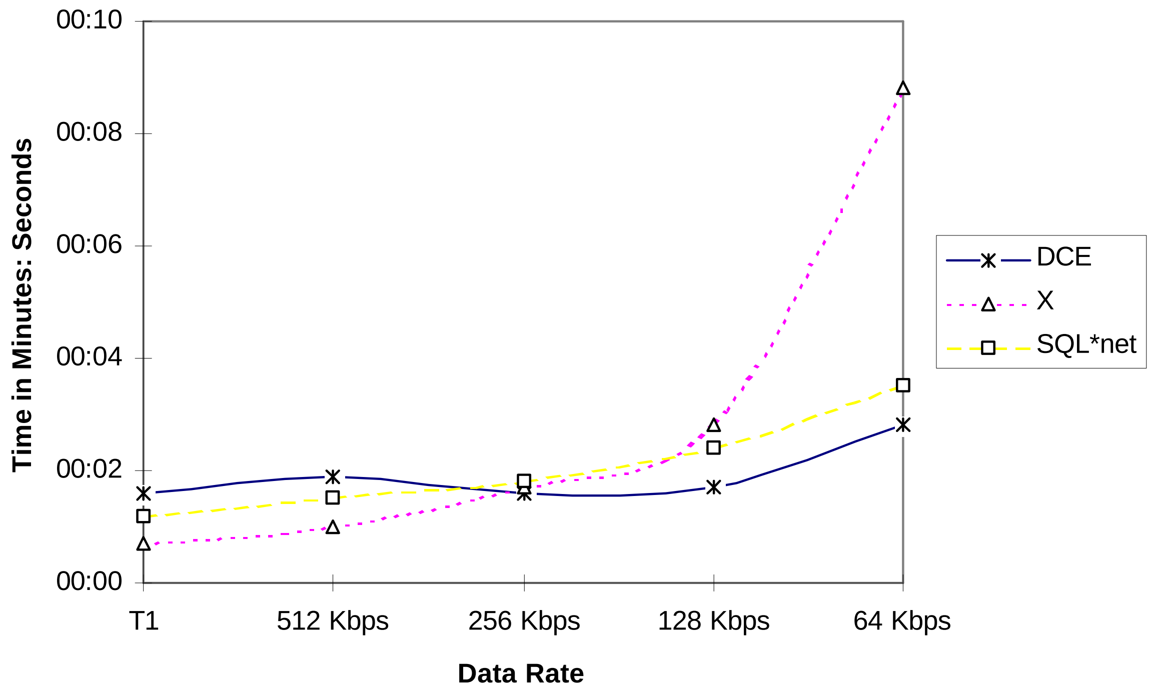


*Figure 2-17: Comparison of DCE, X and SQL*NET for Addition of an Air Carrier (Display JDIC4).*

### 2.5.4 Test Conclusions

For the test cases that involved querying the database and retrieving large amounts of data, the All GEOLOCS and the GEOLOCS in Germany queries, DCE and X perform much better than SQL*NET. For these two queries, the DCE-based application s timings are based upon the sum of querying the initial page of data and switching between pages in order to provide a fair comparison between DCE, X, and SQL*NET. X and SQL*NET do not use a paging algorithm and, therefore, those test cases do not accrue additional time for paging. Also, none of the DCE timings take into account the time to make the initial RPC needed for the CDS to access the correct database. After the initial RPC, the information from this call is cached in memory and does not need to be executed again. Therefore, this time is an initial cost to start the DCE-based application but does not influence the performance of the database operations used in this testing and is not included in the testing results for DCE. The delay in making the initial RPC call was approximately three to four seconds. Beginning at 512 Kbps, SQL*NET performed noticeably worse than both X and DCE as the link capacity was reduced. X s and DCE s performance results were not effected by reducing the link capacity. Also, no noticeable differences in performance existed between DCE and X in the queries. X and DCE perform better at lower link capacity because all data retrieved from the query was not transmitted across the network. SQL*NET has decreased performance at lower link capacities due to its property of retrieving all the data in the query at one time and transmitting all the data across the network. In contrast to SQL*NET, X only transmits the data necessary to paint the user's screen, and DCE uses the paging algorithm previously described in Paragraph 2.3.2.2 to only transmit one page of data upon a user request across the network. Therefore, SQL*NET is slower than X or DCE for any large queries at lower link speeds due to the greater amount of traffic generated on the network.

For the addition of an air carrier, DCE and SQL*NET have approximately the same performance which is better performance than X for lower link capacities. Since little data is sent on the network for this database operation, DCE and SQL*NET do not have significantly decreased performance at lower link speeds. However, X must send information across the network for screen painting, which results in slower performance at lower bandwidths because of the larger amount of data required for screen painting in contrast to the little information sent for the database operation. Since the display is local to the client application for DCE and SQL*NET tests, only information from the database operation is sent across the network. Screen painting information is provided locally from the client for the DCE and SQL*NET cases. In summary, DCE performs as well or better than X or SQL*NET in all test cases, outperforming SQL*NET when executing large queries and outperforming X in database transactions such as adding a carrier.

## 2.6 CONSTRAINTS AND LIMITATIONS

This section provides an overview of constraints and limitations of the products used in developing the DCE prototype.

### 2.6.1 DCE and Encina Constraints and Limitations

Constraints and limitations derived from the prototyping effort are generally administrative or programmatic in nature and address usability, maintainability, and availability requirements. Overall, the DCE and Encina environment is inherently difficult to administer, as is any distributed system that uses a significant number of synchronous and asynchronous processing components that access multiple objects within a networked environment. The DCE and Encina products use standard command-line and interactive script interfaces to administer the cell directory service, security service, distributed time service, and Encina monitors. Transarc's DCE product provides a number of value-added utilities and scripts not provided with OSF's DCE to support product installation, configuration, reconfiguration, DCE backup, and DCE restoration thus providing for slightly easier maintenance and more development productivity. The Encina transaction management product further compensates for DCE administrative and functional deficiencies by providing additional value-added utilities and scripts for administration, fault tolerance logic, and functional capabilities not supported by DCE. Graphical administrative capabilities are provided by subsequent versions of Encina as well as products from third-party vendors.

Additional deficiencies are associated with system modifications during DCE and Encina installation and configuration including the inability to view some ACLs and CDS entries, and the lack of DCE and Encina log administration. Although most of the installed DCE and Encina software is contained within a single directory tree in the system, installation and configuration does modify other UNIX system directories which might cause a problem in a COE environment. The DCE and Encina documentation also does not provide a complete reference of these modifications.

Access control lists and CDS entries cannot be viewed without knowing the reference object. The clearinghouse can be viewed without difficulty, but does not contain pseudo CDS entries for server interfaces or entries for Encina objects (servers) below specific junction points in the Encina directory tree. In order to view these ACLs or CDS entries, the administrator must note the object names when they are created or view the registry.

DCE and Encina log administration is also deficient in the existing system. In the event that a process or daemon inadvertently fails, error and advisory information continues to be logged to the process or server log files indefinitely, thereby saturating the disk partition. This may result in system inaccessibility or resource conflicts for other users.

Reconfiguration of the DCE and Encina monitor cell is not possible without interruption of system operation. The DCE and Encina daemons and processes must be terminated prior to any installation, including product upgrades, or reconfiguration of the cell. If graceful degradation of the system is not possible, there may be residual intermediate system states that may require complete reconfiguration or reinstallation to clear out local databases used only by DCE or Encina. This appears to be primarily a problem with DCE and using the DCE un-configure and un-install utilities sometimes resolves the problem.

Additional fault tolerance in the DCE product is essential in order to fully comply with system availability and reliability requirements. If any DCE daemons or processes inadvertently fail, the system should attempt to determine the cause for failure, perform any corrective action, and attempt to restart the daemons or processes. Administrative intervention is required to determine the cause for failure and to resolve the problem. The Encina transaction management product does attempt to restart Encina mond and PA processes automatically, thus providing additional fault tolerance over DCE.

Programmatic constraints and limitations address implementation and development issues that may affect overall application server architecture and design, and system performance. Currently, Encina requires that a resource manager be specifically allocated to an Encina node manager. This is accomplished by providing Encina with an open string specification that includes the resource manager account name and password. The account name and password specification can be avoided in the open string if the resource manager view and account environment is configured properly. The use of this open string specification prevents any application server from accessing any resource manager account. In addressing the issue, the database vendor needs to support an XA interface to be used in multi-tasking, and Encina needs to provide login capabilities by the resource manager.

Encina only provides for the creation of a finite number of PAs. In order to accommodate future system availability requirements, a large number of processing agents must be created. However, the processing agents are created when the application server is initialized, and are not dynamically created as needed. This requires additional system resources to administer and maintain processing agents not in use. It would be more feasible to dynamically create them as needed.

Other Encina deficiencies include constraints regarding the server environment specifications, Encina libraries, and Encina include files. Encina requires that the environment for the server be set in the server software instead of defaulting to the shell environment of the parent process. Encina libraries are also highly order dependent when attempting to link application binaries; library definitions specified earlier can be overwritten later and vice-versa. Also, the Pro*C compiler has difficulty parsing some Encina include files which are written in ANSI C.

## 2.7 FUTURE MIGRATION STRATEGIES

The DCE prototype provided significant insight into the overall incorporation and implementation of DCE and transaction management capabilities into DII database applications. Strategies regarding migration of an existing two-tiered application to a three-tiered architecture and incorporation of the DCE infrastructure and transaction management capabilities into DII applications are discussed from a top-level perspective. The migration strategy should focus on reusability of applicable plans, documents, operational environment, and applications to minimize cost and impact. The strategy addresses general issues and a specific migration methodology employed to ensure a successful migration of the applications. The software migration strategy is delineated in Paragraph 2.3.1.3. Other activities include a DCE pilot project and definition of the overall DCE and Encina monitor cell architecture and naming conventions.

General issues that encompass system level technical and management activities should address the following:

- Developmental and Operational Environment. Identify the existing DCE common operating environment including platforms, support products, and development environment. This will provide the basis for incorporating the DCE and Encina infrastructure in order to ensure product compatibility and interoperability with the existing infrastructure and products.

- Concept of Operations (CONOPS). The existing operational concepts, environment, and infrastructure at all GCCS sites will provide the basis for developing a new CONOPS with the DCE and Encina infrastructure.

- GCCS System Requirements. Review and update the existing GCCS system and software requirements for incorporation of DCE and transaction management capabilities.

- GCCS System Architecture. Review and update the existing GCCS system architecture for incorporation of DCE and transaction management requirements and configuration of the DCE and Encina monitor cells. Partitioning the applications to accommodate a three-tiered architecture may require a redesign effort. Interoperability and interdependency between sites using and not using DCE should be addressed Separate operability should pose no known problems if there are no common interfaces (e.g., clients and servers implemented using SUN RPCs can operate independently of clients and servers implemented using DCE RPCs). However, it may not be possible to combine both environments within a single client/server scenario.

- GCCS System Interfaces. Review and update the existing GCCS system and software interface specification to accommodate the new DCE and transaction management infrastructure. The interface specification should delineate interface definitions for inter- and intra- DCE Cell components and GCCS objects.

- GCCS Applications and Operational Sites. Identify GCCS target applications and sites for migration to DCE and Encina. The migration strategy should delineate procedures that provide a smooth transition to the DCE environment with minimal loss of functionality at the operational sites and consideration to redundant hardware and software.

- DCE and Encina Infrastructure. Adopt and develop a plan to incorporate the DCE and Encina infrastructure into the existing developmental and operational sites with minimal loss of existing functionality.

- DCE Products. DCE, transaction management, and support products should be identified that provide the most cost effective solution with conformance to the DII COE. Requirements of these products include: open systems compatibility, potential for future performance and capacity growth, lifecycle costs, operating environment constraints, interoperability with other products, tools, and GCCS components, portability, and increased maintainability to the applications being developed. Vendors should be accountable for ensuring compatibility with other products.

- Management and Implementation Plans. Identify and modify new or existing management and implementation plans for incorporating DCE and transaction management products into the GCCS system. Provide for identification and incorporation of DCE into existing management and administrative policies, procedures, and plans for DCE migration.

- Redevelopment Effort. The incorporation of DCE and transaction management requirements into GCCS may require significant redesign and redevelopment of existing applications depending upon modularity and coupling of the existing design, product support, new requirements, and concept of operations.

- Training and Education. Additional technical education and skill development in both DCE and Encina is necessary to develop, deploy, and administrate the system.

### 2.7.1 DCE Pilot Project

Additional DCE activities should be considered to further confirm the feasibility of migrating the DII applications to DCE. A DCE pilot project will incorporate the remaining threads of the GCCS S&M application into the three-tiered client/server architecture using DCE and Encina. The same iterative, rapid application development technique will be used for this functionality as with the Add Carrier thread developed for the DCE prototype. In redeveloping the application, a three-tiered client/server development tool will be selected that satisfies DCE and future requirements for the GUI. An additional phase of the pilot project will provide for investigation, analysis, and selection of a three-tiered client/server development tool that incorporates a DCE-compliant GUI generator for inclusion in the DII COE.

The resulting application developed in the pilot project should be deployed at one or more specific operational sites in order to verify infrastructure and application functionality using DCE and Encina. This implies that DCE and application functional redundancy discussed in other sections of this document will need to be incorporated to ensure continuous functionality of the operational site. Proposed target operational sites include TRANSCOM or one of its components such as ACC. Both of these sites are database server sites and can function as DCE and Encina monitor cells consistent with the proposed DCE cell architecture outlined in the following section. Since the prototype application was developed using the DCE 1.0.3a and Encina 1.1 products that support Solaris 2.3 UNIX platforms existing at the operational sites, upgrading the prototype applications to use DCE 1.1 and Encina 2.0 in support of Solaris 2.4 will not be necessary. Both DCE and existing applications can function independently at the same site without losing operational functionality. Prior to deployment, DCE and Encina segmentation requirements will be assessed and incorporated into the pilot project development activities.

Additional benefits of the pilot project include validation of the *GCCS DCE Implementation Plan* and *DCE Application Programmer's Guide* produced by the DCWG and lessons learned from the development, implementation, segmentation, deployment, and administration activities associated with the pilot project.

### 2.7.2 Proposed GCCS Cell Topology

Currently, there are approximately 25 application sites and 16 resource manager sites within GCCS. Although the DCE cell can be specified to any size and any configuration, the overall GCCS architecture will incorporate multiple local area network clusters communicating over multiple wide-area networks. The fundamental justification for the GCCS DCE cell architecture stipulates that clients and servers communicating primarily with each other be grouped into a single cell, thereby reducing intra-cell communication data traffic and processing overhead, and simplifying cell administration. This infers that DCE cells should be concentrated around resource managers and also incorporate application server sites that primarily communicate with that resource manager. This would indicate that one or more sites or organizations would need to share system and personnel resources in administration of the DCE cell. Table 2-7 illustrates a possible GCCS DCE cell architecture. Proposed DCE cell names coincide with the resource manager site names. Additionally, the Encina monitor cell architectures coincide with each DCE cell or multiple monitor cells or could reside within each DCE cell, depending upon administrative responsibilities and security issues.

It is recommended that each DCE cell incorporate adequate redundancy for security and cell directory service in order to comply with availability and reliability requirements. If a particular DCE cell boundary encompasses multiple application server sites, redundancy can span across sites ensuring functionality in the event of a site failure. Likewise, in the event of DCE cell failure, it is recommended that application server and resource manager functional redundancy be contained within an alternate DCE cell.

The existing site, organizational, administrative and security infrastructure can remain intact. DCE cell administration is not easily integrated with system administration and should therefore remain a separate responsibility. If a site does not currently have any administrative or security infrastructure, then the other site(s) within the DCE cell can absorb the administrative responsibilities.

*Table 2.7: Premliminary GCCS/DCE Cell Architecture.*

| SITE | DCE CELL | DCE CELL NAME | ENCINA CELL NAME |
|---|---|---|---|
| CENTAF | ACC | /…/gcc.smil/acc | centaf |
| LANTFLT | ACOM | /…/gcc.smil/acom | acom |
| MARFORLNT | | | |
| AFMC | AMC | /…/gcc.smil/amc | amc |
| SPACECOM | | | |
| MARFORPAC | ARPAC | /…/gcc.smil/arpac | arpac |
| PACOM | | | |
| NAVCENT REAR | CENTCOM | /…/gcc.smil/centcom | centcom |
| NAVCENT FORWARD | | | |
| AREUR | EUCOM | /…/gcc.smil/eucom | eucom |
| ARCENT | FORSCOM | /…/gcc.smil/forscom | forscom |
| ANMCC | NMCC | /…/gcc.smil/nmcc | nmcc |
| CNO | | | |
| HQUSAF | | | |
| HQUSMC | | | |
| PACFLT | PACAF | /…/gcc.smil/pacaf | pacaf |
| USASOC | SOCOM | /…/gcc.smil/socom | socom |
| MSC | TRANSCOM | /…/gcc.smil/transcom | transcom |

| SITE | DCE CELL | DCE CELL NAME | ENCINA CELL NAME |
|---|---|---|---|
| MTMC | | | |
| STRATCOM | TRANSCOM | /…/gcc.smil/transcom | transcom |
| NAVEUR | USAFE | /…/gcc.smil/usafe | usafe |
| USFK (YONGSAN) | USFK (TAEGU) | /…/gcc.smil/usfk | usfk |
| HQDA | HQDA | /…/gcc.smil/hqda | hqda |
| SOUTHCOM | SOUTHCOM | /…/gcc.smil/southcom | southcom |

Table 2-7 delineates a new proposed cell architecture based on RM (database) sites. The cell architecture ***is only preliminary*** and does not encompass all sites. Encina monitor cells coincide with the DCE cells for simplification. Cell names are derived from the resource manager sites. System administration will remain constant for each site. DCE cell administrative responsibilities will be associated with the RM or database site, and security administration will need to be coordinated among sites within a particular DCE cell.

## 2.8 CONCLUSIONS

The DII DCE prototyping effort provided an opportunity to investigate the implications of incorporating the OSF DCE middleware component and a transaction manager into a DII application. It validates the ability to migrate the DII application from the existing two-tiered architecture into a three-tiered client/server architecture. Transarc's DCE Version 1.0.3a and Encina Version 1.1 products were used to implement the DCE prototype and were selected according to requirements that included compliance with open systems standards, support for the UNIX environment, product interoperability, functional and non-functional capabilities, and consistency with the DII COE.

Transarc's DCE is a middleware product ported from OSF's DCE that provides a consistent interface infrastructure for developing distributed client/server applications, thus achieving compatibility and interoperability between software and hardware components. It provides a number of services to ensure transparent and secure access to objects, and synchronization of the operating environment through the CDS, security service, and DTS. It also allows for greater distribution and scalability of processing components, thereby ensuring higher performance and resource manager availability, and reducing data access times.

Although all DCE products are based on OSF'S DCE, Transarc has developed a number of additional value-added utilities that simplify DCE application development and administration for UNIX-based systems. DCE alone does not provide transaction management capabilities; a separate DCE-compliant transaction processing system will need to be developed or purchased that satisfies any transaction management requirements. Transarc's Encina provides a transaction processing system that is built using the Transarc DCE infrastructure. Encina simplifies administration of clients, servers, and resource managers and facilitates an increase in productivity by reducing the number of Application Program Interface (API) calls and DCE interfaces necessary to develop clients and application servers. Additional tracing and debugging utilities are provided with Encina that are not inherent in the DCE product.

The DCE prototype effort provided insight into DCE migration activities for the GCCS Scheduling and Movement application. Subsequent system analysis and design activities determined which components should incorporate DCE-only, Encina-only, or a combination of both DCE and Encina clients and servers. In migrating the applications, it was imperative that the complete DCE, Encina, GUI software and hardware development environments and infrastructure be in place prior to commencement of any development activities. An iterative, rapid application development approach was used in the prototyping effort that incorporates event and object modelingtechniques. This approach focused on software and hardware reusability, incremental analysis, development, and testing, and is recommended for other migration activities. Automated tools and utilities to build applications, configure, and administer the DCE and Encina environment were developed to simplify development and administration and increase productivity.

The DCE environment is inherently difficult to administer, as is any distributed system that uses a significant number of synchronous and asynchronous processing components that access multiple objects within a networked environment. The degree of usability provides for simplicity in accessing component, configuration, and local DCE and transaction manager database information. Usability and simplicity of DCE and Encina administration intersect and influence maintainability, availability, and reliability of the overall system and should be considered in overall system lifecycle costs. The DCE and Encina products incorporated in the DCE prototype use standard command-line and interactive script interfaces to administer the cell directory service, security service, distributed time service, recoverable queuing service, and Encina monitors. A later version of Encina provides graphical administrative capabilities that simplify administration. Other products will need to be adopted to simplify DCE administration.

Additional DCE activities should be considered to further confirm the feasibility of migrating the DII applications to DCE. A DCE pilot project will incorporate DCE and Encina into the remaining GCCS S&M threads and provide for investigation, analysis, and selection of a three-tiered client/server development tool. Target operational sites were identified as database server sites that can function as DCE and Encina monitor cells consistent with the proposed DCE cell architecture.

Successful migration of the DII applications from a two to a three-tiered client/server architecture requires careful analysis of the existing system functionality, incorporation of new requirements, and implementation planning. Program and technical risk can be mitigated by defining and controlling the scope of effort, evaluating technological impacts, enforcing quality control, and adhering to delivery schedules. Component reusability, development simplification, and the availability of inter-operable COTS products will significantly reduce overall system lifecycle costs. Adequate redundancy of system functionality and components including clients, application servers, DCE files and databases, and resource managers (database servers) is required in order to comply with system availability, reliability, and performance requirements.

## APPENDIX A

## ACRONYMS

# APPENDIX A - ACRONYMS

| | |
|---|---|
| ACL | Access Control List |
| API | Application Program Interface |
| CDS | Cell Directory Service |
| COE | Common Operating Environment |
| CONOPS | Concept of Operations |
| CORBA | Common Object Request Broker Architecture |
| COTS | Commercial Off-the-Shelf |
| CSU | Channel Service Unit |
| DBMS | Database Management System |
| DCE | Distributed Computing Environment |
| DCWG | Distributed Computing Working Group |
| DII | Defense Infrastructure Initiative |
| DSU | Data Service Unit |
| DTS | Distributed Time Service |
| ecm | Encina Cell Manager |
| enm | Encina Node Manager |
| 3GL | Third Generation Language |
| 4GL | Fourth Generation Language |
| GCCS | Global Command and Control System |
| GEOLOC | Geographic Location |
| GUI | Graphical User Interface |
| IDL | Interface Definition Language |
| I/O | Input/Output |
| JOPES | Joint Operations Planning and Execution System |
| kbps | Kilobits Per Second |
| LAN | Local Area Network |
| Mbps | Megabits Per Second |
| mond | Monitor Daemon |
| NPG | Non-Unit Personnel Generator |
| OCI | Oracle Call Interface |
| ODBC | Open Database Connectivity |
| ODS | OPLAN Directory Service |

| | |
|---|---|
| OIS | Objective Interface Systems |
| OPLAN | Operational Plan |
| OS | Operating System |
| OSF | Open Software Foundation |
| | |
| PA | Processing Agent |
| PAX | Passenger |
| | |
| RDA | Requirements Data Analysis |
| RDBMS | Relational Database Management System |
| RepSpecs | Representational Specifications |
| RM | Resource Manager |
| RPC | Remote Procedure Call |
| RQS | Recoverable Queuing Service |
| | |
| S&M | Scheduling and Movement |
| SFS | Shared File System |
| SQL | Structured Query Language |
| SRA | Systems Research & Applications |
| SRS | Software Requirements Specification |
| | |
| TDS | Transaction Distribution Service |
| TIDL | Transactional Interface Definition Language |
| tpm | Transaction Processing Monitor |
| TRPC | Transactional Remote Procedure Call |
| TX | Transaction |
| | |
| VADS | Verdix Ada Development System |
| | |
| WAN | Wide Area Network |
| WWMCCS | Worldwide Military Command and Control System |

## APPENDIX B

## BIBLIOGRAPHY

# APPENDIX B - BIBLIOGRAPHY

## B.1 SPECIFICATIONS

- *Software Requirements Specification for Scheduling and Movement Client Server*. Systems Research and Applications Corporation. Contract No. DCA100-91-C-0123, November 1993.

## B.2 OTHER PUBLICATIONS

- *OSF DCE Administration Guide - Introduction*. Open Software Foundation. 1993.
- *OSF DCE Application Development Guide*. Open Software Foundation. 1993.
- *OSF DCE Application Development Reference*. Open Software Foundation. 1993.
- *OSF DCE User's Guide and Reference*. Open Software Foundation. 1993.
- *Oracle7 for Sun SPARC Solaris 2.x Installation & Configuration Guide*. Oracle Corporation. 1994.
- *Oracle7 Server for UNIX Administrator's Reference Guide*. Oracle Corporation. 1992.
- *Oracle7 Server Messages and Codes Manual*. Oracle Corporation. 1992.
- *Oracle7 Server SQL Language Reference Manual*. Oracle Corporation. 1992.
- *Pro*C Supplement to the Oracle Precompilers Guide*. Oracle Corporation. 1992.
- *Programmer's Guide to the Oracle Call Interfaces*. Oracle Corporation. 1992.
- *Programmer's Guide to the Oracle Pro*C Precompiler*. Oracle Corporation. 1992.
- Rosenberry, Ward. *Understanding DCE*. 1992.
- Shirley, John. *Guide to Writing DCE Applications*. 1992.
- *DCE Administration Guide - Introduction*. Transarc Corporation. 1994.
- *DCE Administration Reference*. Transarc Corporation. 1994.
- *DCE Command Reference*. Transarc Corporation. 1994.
- *DCE Installation and Configuration Guide*. Transarc Corporation. 1994.
- *Encina Installation and Configuration Guide*. Transarc Corporation. 1994.

- *Encina Monitor System Administrator's Guide and Reference*. Transarc Corporation. 1994.
- *Encina SFS System Administrator's Guide and Reference*. Transarc Corporation. 1994.
- *Encina Transactional-C Programmer's Guide and Reference*. Transarc Corporation. 1994.
- *Introduction to DCE*. Transarc Corporation. 1994.
- *Toolkit System Administrator's Guide and Reference*. Transarc Corporation. 1994.

## APPENDIX C

## GRAPHICAL USER INTERFACE (GUI) PRODUCTS

# APPENDIX C - GRAPHICAL USER INTERFACE (GUI) PRODUCTS

This appendix presents a table of GUI products that have been succesfully integrated with Encina 1.1 as provided by Transarc Corporation. This listing should provide an initial candidate list of GUI products to be used in the development and migration of database applications as described in Paragraph 2.4.2.3.

*Table C-1: Graphical User Interface (GUI) Products.*

| Product Name | Vendor | Platform Support |
|---|---|---|
| EncinaBuilder | Transarc Corp. | SUN Solaris 2.4 (2Q)<br>Windows 3.1/NT |
| Entera | Open Environment Corp. | SUN Solaris 2.3<br>Windows 3.1/NT |
| Forte Development Systems | Forte Software, Inc. | SUN Solaris 2.3, 2.4<br>Windows 3.1/NT/95 |
| JAM7/TPi | JYACC, Inc. | SUN Solaris 2.3<br>Windows 3.1/95 |
| PowerBuilder | Powersoft Corp. | SUN Solaris 2.4<br>Windows 3.1/NT |
| Seer*HPS | Seer Technologies | SUN Solaris<br>Windows NT |
| XVT Software | XVT Software, Inc. | SUN Solaris 2.4<br>Windows 3.1/NT/95 |

## APPENDIX D

## PERFORMANCE TESTING DATA

| [illegible] | | | | | | | |
|---|---|---|---|---|---|---|---|
| | Local | 10 Mbps | T1 | 512 Kbps | 256 Kbps | 128 Kbps | 64 Kbps |
| DCE | [illegible] | [illegible] | [illegible] | [illegible] | [illegible] | [illegible] | [illegible] |
| DCE | [illegible] | [illegible] | [illegible] | [illegible] | [illegible] | [illegible] | [illegible] |
| X | [illegible] | [illegible] | [illegible] | [illegible] | [illegible] | [illegible] | [illegible] |
| X | [illegible] | [illegible] | [illegible] | [illegible] | [illegible] | [illegible] | [illegible] |
| SQLnet | [illegible] | [illegible] | [illegible] | [illegible] | [illegible] | [illegible] | [illegible] |
| SQLnet | [illegible] | [illegible] | [illegible] | [illegible] | [illegible] | 00:11.1 | 00:02.0 |

## APPENDIX D - PERFORMANCE TESTING DATA

This appendix presents the raw numerical data associated with the performance testing data gathered by SRA. All data is given in the form of minutes and seconds with seconds given to the accuracy of tenths of seconds. Each of the boxes displayed below relate directly to the chart with the same title which can be found in Paragraph 2.5.

| ADD Carrier - DISPLAY JDIC4, DATABASE JDIC3 | | | | | |
|---|---|---|---|---|---|
| | | | | | |
| | T1 | 512 Kbps | 256 Kbps | 128 Kbps | 64 Kbps |
| DCE | 00:01.6 | 00:01.9 | 00:01.6 | 00:01.7 | 00:02.8 |
| X | 00:00.7 | 00:01.0 | 00:01.7 | 00:02.8 | 00:08.8 |
| SQLnet | 00:01.2 | 00:01.5 | 00:01.8 | 00:02.4 | 00:03.5 |